# AN EMPIRICAL INVESTIGATION OF HALSTEAD'S SOFTWARE LENGTH FORMULA

by

CHERN-HWANG HWANG

B.A., Tunghai University, 1977
M.S., Kansas State University, 1984

---

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1988

Approved by:

Major Professor

# ACKNOWLEDGEMENTS

I wish to express my sincere appreciation to my major professor Dr. D. Gustafson for his aid in the selection of this subject and for his patience and guidance with this research.

Thanks is also expressed to Dr. S. McNulty and Dr. B. Hankley for their valuable suggestions and comments on this report.

## Table of Contents

# INTRODUCTION

One of the most fundamental problems of software engineering today - both in theory and in practice - is the prediction of the software length. Many studies have indicated the length of the program is consistently correlated with some other complexity measurements of program's characteristics [Basili 83], [Mata 84], [Jensen 85], [Ivan 87]. In addition, a recent survey on software economics has listed software length estimation as the first major issue needing further research [Bohem 84]. The software length can be measured from a number of aspects, such as, line of source codes [Dijkstra 72], executable statements [Curtis 79] or total number tokens in the program [Halstead 72]. In 1972, Halstead proposed a simple formula for predicting the length of a program based on the number of unique operators and operands used by the program [Halstead 72]. Christensen has stated the general advantages of this operator/operand approach as: [Chris 81]

- An explainable methodology for calibrating a measurement instrument.
- A more nearly universal measure, since the approach is consistent across the boundaries of programming languages.
- The ability to relate some of the effects of programming style to measure quantities.

Halstead established a theory based on these empirical findings, and extended it into various metrics for measuring the characteristics of the software in his literary work [Halstead 77]. This landmark work is well known as software science.

**The Models Of Software Length Measurement**

The estimated length suggested in the software science is simply a function of the number of unique operators ($\eta_1$) and operands ($\eta_2$)

$$\hat{N} = \eta_1 \cdot \log_2(\eta_1) + \eta_2 \cdot \log_2(\eta_2). \quad (1)$$

Halstead's formula has been widely applied. However, it is also seriously questioned about its inaccuracy [Smith 80],[Lassez 81], [Shen 83], [Hamer 82], and ambiguity [Elshoff 78],[Lassez 81], [Shen 83], [Fitsos 80]. The one limitation of the length formula (1) as a tool for estimating program length is that $\eta_1$ only can be evaluated after the program has been written. Fitsos proposed using a model that only depends on operand vocabulary size [Fitsos 80],

$$\hat{N} = c + \eta_2 \cdot \log_2(\eta_2) \quad (2)$$

where $c$ is a language dependent constant. His suggestion was based on the observation of 490 PL/S modules and Elshoff's data for 34 PL/I modules [Elshoff 78]. Fitsos concludes that the number of distinct operators ($\eta_1$) for programs written in a higher level language tends to be a constant. In other words, the program length can be determined by a function of distinct operands number ($\eta_2$). This hypothesis was reaffirmed later by Christensen [Chris 81]. Fitsos's methodology breaks the restrictions of Formula (1), since the estimating process can be conducted in the variables declaration section of the program. Formula (2) was extended by Albrecht [Albrecht 83], who reports the data for 14 modules, and suggests an alternative model of the form

$$\hat{N} = c \cdot \eta_2 \cdot \log_2(\eta_2) \quad (3)$$

Formula (2) and (3) have been investigated by Levitin [Levitin 85]. The results of the Levitin's experiments indicated that the estimator (3) is superior to the estimator (2). About the same time, [Jensen 85] studied the software measures for real-time programs, and proposed a length estimation equation

$$\hat{N} = \log_2(\eta_1!) + \log_2(\eta_2!) \qquad (4)$$

Jensen found that the estimation results of Formula (4) are more precise than that of Halstead's on his data set.

**The Nature of The Problem**

Models described in the previous section are based upon various assumptions. For example, Halstead divides a program of length $N$ into $N/\eta$ substrings of length $\eta$ (which is the sum of number of distinct operands and operators), and assumes there are no duplications of these substrings. He also assumes that operators and operands alternate in the program [Halstead 77]. In the models of (2) and (3), the assumptions are based on the number of operators being a constant, and they also employ the portion of Halstead's formula, $\eta_2 \cdot \log_2(\eta_2)$, to determine the value for the operands. Jensen [Jensen 85] did not mention any assumptions nor deriving process of the equation (4).

The above assumptions are not always true in real programming environments. For instance, the operators and operands do not necessarily alternate. The statement " *for(;;)* { " is allowed in the language C. There are four operators occurring consecutively, namely "for","(",";", and "{". Regarding to formulae (2) and (3), the operators behave as a constant for large programs. Fitsos [Fitsos 80] and Albrecht [Albrecht 83] both agree that the term

$\eta_2 \cdot \log_2(\eta_2)$ can determine the total value for operands in the program. However, on observing the data sets used in their research, the author found that the two terms in Halstead's formula can not be used to estimate $N1$ and $N2$ respectively; that is, $\eta_1 \cdot \log_2(\eta_1)$ was not a good estimator of $N_1$, and $\eta_2 \cdot \log_2(\eta_2)$ was not a good estimator of $N_2$.

**The Aims of The Study**

In this report, the models are developed based upon the data sets and without unnatural assumptions are introduced. Three different data sets ( UNIX source codes, C programs written in the course CMPSC 541, and Pascal programs ) are used to investigate the estimation models. A correlation analysis between the estimated and the actual length is presented. Additionally, the relative error is used for comparing the accuracy of the estimations.

# SOFTWARE SCIENCE AND LENGTH ESTIMATION

The ever-increasing cost of program development has made the measurement of software characteristics more important than it has ever been before. Software science includes some of the most often used measures. The metrics proposed by Halstead's software science are briefly discussed in this chapter. Several articles relating to the software science length estimation are reviewed.

## The theory of software science

Halstead's software science is widely recognized as an important analytical tool for the analysis and design of software. Halstead argues that algorithms or programs have measurable characteristics analogous to the characteristics, such as mass, that are used in physical laws. He also suggests that a set of useful measures of program characteristics can be derived from a count of the number and the frequency of distinct operators and operands in an algorithm or a program. The basic counts of software science are:

$\eta_1$ = number of distinct operators

$\eta_2$ = number of distinct operands

$N_1$ = number of operator occurrences

$N_2$ = number of operand occurrences

Followings are the program properties measurements proposed by Halstead in the software science:

*Program length*

Program length $N$ is defined as the sum of the total number of operators $N_1$ and the total number of operands $N_2$ (ie. $N = N_1 + N_2$). The value of $N$ can be approximated by an estimator $\hat{N}$ that is defined as:

$$\hat{N} = \eta_1 \cdot \log_2(\eta_1) + \eta_2 \cdot \log_2(\eta_2).$$

*Program volume*

A program volume metric $V$ is defined as

$$V = N \cdot \log_2(\eta)$$

Volume, in the other sense, is the size of an implementation, which can be thought as the number of bits necessary to express an algorithm.

*Potential volume*

The potential volume $V^*$ is the minimum possible volume for the given algorithm. $V^*$ is of the form

$$V^* = (2 + \eta_2^*) \cdot \log_2(2 + \eta_2^*),$$

where $\eta_2^*$ is the observed input operands required by the program.

*Program level (difficulty)*

Any given algorithm with volume V is considered to be implemented at the program level L, which is defined as

$$L = \frac{V^*}{V},$$

and the inverse of the program level is termed the *difficulty*. That is

$$D = \frac{1}{L}.$$

*Program effort*

Program development requires more effort when the size of the program increases; but, it needs less effort when the language is high. The effort E, then, derived as:

$$E = \frac{V}{L}.$$

The unit of measurement of E is "elementary mental discriminations".

*Programming time*

The programming time T is proportional to the effort E in developing a program, E is defined as the form

$$T = \frac{E}{S},$$

for some constant $S$. The constant $S$ represents the speed of programming. In other word, the number of mental discriminations per second of which programmer is capable. An $S$ value of 18 is normally is normally used. This number is based on the work of Stroud.

Software science has been accepted and discussed by many authors. There are many valuable and important articles concerned with software science that are listed and annotated in bibliography [Leslie 87].

**Program Length Estimation**

In software science, the length of a program is a function of the number of unique operators and operands. This hypothesis has received the most attention since it can be easily tested. Halstead assumed that the accuracy of the formula is dependent on the "purity" of the algorithm implementations. The types of 'impurity' can be classified, according to [Halstead 77], as

- a complementary operation,
- ambiguous operands,
- synonymous operands,
- common subexpression,
- unwarranted assignment, and
- unfactored expressions.

[Elshoff 78] measured 154 programs and confirmed this hypothesis, he also pointed out that " if $N \approx \hat{N}$ only for pure or well programmed algorithms, then a simple check for pure or well programmed programs is available. ".

The operators/operands can be viewed as simply analogous to the daily conversational sentence. Operators are the verbs, and operands are subjects or objects. However, in some programming language, the classification of operator and operand becomes very ambiguious. Most of the supporting experiments presented in the [Halstead 77] derived from the collected algorithms of the ACM and very small program in ALGOL and FORTRAN. In both languages, it is not difficult to classify a token into operator or operand. However, in other languages, sometimes, it can lead to an ambiguous situation. Nevertheless, from the aspect of length estimation, [Shen 83] Shen pointed out the misclassification of any token has virtually no effect on the final estimate, since

$$\hat{N} = \eta_1 \cdot \log_2 \eta_1 + \eta_2 \cdot \log_2 \eta_2 \approx \eta \cdot \log_2 \frac{\eta}{2.}$$

However, except for length estimation, when the other characteristics are concerned, Lassez criticized that the software science is not applicable because of unclear definitions of operator, operand and input/output parameter [Lassez 81].

The tokens in the declaration sections are not counted as the part of length of the program [Halstead 77]. It causes an obvious variations of estimation, since the variable declaration sections in some languages (eg,. data division in Cobol) represent a significant portion of the programming effort. Therefore, many authors suggest that all software science analysers should count operatores and operands in declaration sections as well as in procedure sections. [Shen 79], [Fitsoss 79], [Elshoff 78], [Lassez 81]

Experiments have been conducted by Halstead and others to validate the length estimation. Tests have been conducted on FORTRAN programs [Halstead 77], [Basili 83]; Cobol programs [Bulut 74], [Zweben 79], [Shen 79a]; PL/I programs [Elshoff 76], [Smith 80]; Pascal programs [Feuer 79], [Fitz 78], [Lassez 79]; APL modules [Zweben 79]; IBM370 Assembly programs [Smith 80]; and C program [Crawford 85], all observing high correlation between predicted and observed length.

However, some found that Halstead's estimated length tends to be low for large programs and high for small programs [Smith 80], [Fitsos 80], [Shen 79a]. Shen asserted that the Halsteads's length estimation appears to work best for programs of size in the range between 2000 and 4000 [Shen 83]. Feuer also reported that the length equation overestimates the actual length 80% of the time for 197 PL/I programs. In his experiment most of the programs are less than 2000 [Feuer 79]. Therefore, Shen has suggested that the relative error of Halstead length equation can be minimized by dividing a large program into modules of reasonable size and then summing the individual estimates [Shen 83].

Shooman, in 1977, used a set of psychometric relationships suggested by Zipf to estimate program length from the number of unique operators and operands of a program [Shooman 83]. Shooman views the program as a string of tokens. The token string which represents the program is generated by choosing an operator from the operator set randomly, then choosing an operand from the operand set at random, and continuing this alternation process. The process halts whenever the last operator or operand is chosen for the first time. Based on these assumptions, he derives a series equation to estimate the length of program. Mohanty [Mohanty 79] has also demonstrated that a close agreement exists between the software science results and the results obtained by the application of Zipf's law. However, Sooman's work has also been criticized extensively [Moranda 85] on the ground of meaningless substitutions, equating different probability constants, alternation of sourcer data set, and violation of Zipf's law.

# EXPERIMENT DESCRIPTION AND ALTERNATIVE MODELS

The primary methodology applied in the study is based on empirical observation of program files. Three data sets of two languages, C and Pascal, are used to investigate the length estimation models. In order to view the behavioral trend when actual length increases or decreases, the data sets are also divided into various partitions. Four new models are introduced, two of them are built based upon the transformed data so that the models become more appropriate for linear modeling procedures. Another two models are suggested with the same pattern, but with the error terms handled in the different ways. The relative error is used for comparing the superiority or inferiority of the models. Eight models are analyzed and compared using all of the data or the partitioned data sets.

## Counting Rules

In the current research, modules that extract the operators and operands from the programs were implemented (See Appendix D). The rules that distinguish tokens of operators or operands are as follows:

Operators - keywords;

Operators symbols; (a pair symbols is counted as one)

function name;

procedure name;

Operands - variable name;

numerical constant;

quoted string;

Comments are not considered operators nor operands.

**Alternative Models**

Each of the collected data sets have many cases with length less than 500 (See Figure 1). If the data sets were investigated directly without any transformation, the model could produce results much favored to the programs of large size in terms of relative error. Therefore, the logarithmic transformation is applied in order to avoid this situation (See Figure 2). The selection of logarithmic transformation is quite subjective; however, other transformation procedures are also worthy of being studied in the future.

Based on initial analysis efforts, the combinations of $\eta_1$ and $\eta_2$ to be used as independent variables are $(\eta_1 + \eta_2)$ and $(\eta_1 \cdot \eta_2)$. The logrithm of these two combinations are suggested because of their higher correlation with that of $N$ than any others in the preliminary effort. When observing the data displayed on figures, the distribution over the domain of the variable is asymmetric with positive skew (i.e. long tail to the right). The transformation brings the high variability for large programs to be more homogeneous with that of small programs.

# Figure 1

## Frequency Distribution of Raw Data Sets

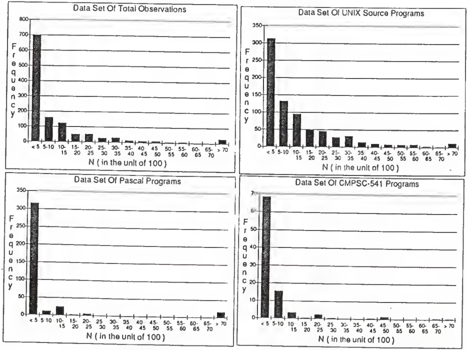
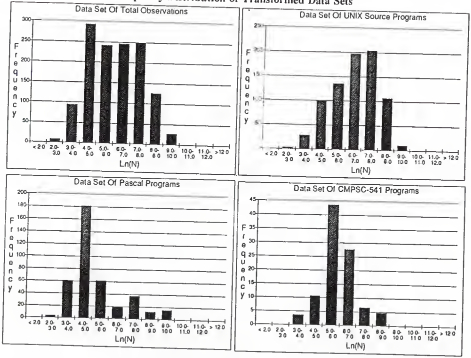


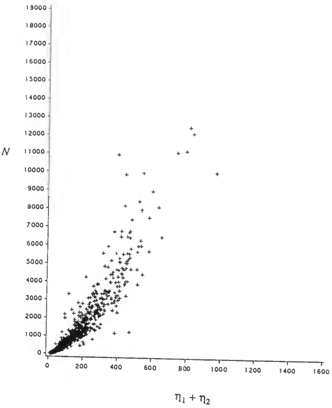
**Figure 2**

**Frequency Distribution of Transformed Data Sets**



# Figure 3

$N$ *vs.* $\eta_1 + \eta_2$



# Figure 4

*N vs.* $\eta_1 \cdot \eta_2$

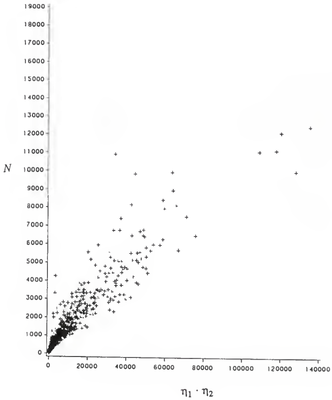


Figure 5 and 6, present the distribution of the data points after the raw data was transformed. The simple linear regression modeling technique, then, is applied to construct the estimating models. The $ln(N)$ value is estimated by the value of $ln(\eta_1+\eta_2)$ and $ln(\eta_1 \cdot \eta_2)$, of the forms

$$ln(N) = \beta_0 + \beta_1 \cdot \ln(\eta_1+\eta_2) + \varepsilon, \tag{5.1}$$

and,

$$ln(N) = \beta_0 + \beta_1 \cdot ln(\eta_1 \cdot \eta_2) + \varepsilon. \tag{5.2}$$

The estimated $N$ value can be obtained from the above equation by applying the inverse transformation. The models are then expressed in the equations:

$$N = exp(\beta_0 + \beta_1 \cdot ln(\eta_1+\eta_2)) \cdot \varepsilon^* \tag{6.1}$$

and,

$$N = exp(\beta_0 + \beta_1 \cdot ln(\eta_1 \cdot \eta_2)) \cdot \varepsilon^* \tag{6.2}$$

where $\varepsilon^* = exp(\varepsilon)$. The equations (6.1) and (6.2) then, are simplified as:

$$N = \gamma \cdot (\eta_1+\eta_2)^{\beta} \cdot \varepsilon^* \tag{7.1}$$

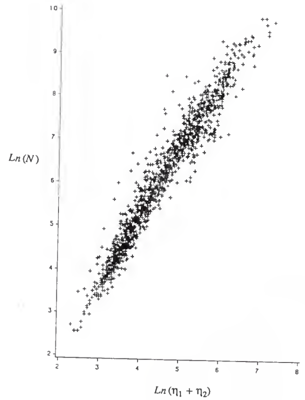
and,

$$N = \gamma \cdot (\eta_1 \cdot \eta_2)^{\beta} \cdot \varepsilon^* \tag{7.2}$$

where $\gamma = exp(\beta_0)$ and $\beta = \beta_1$.

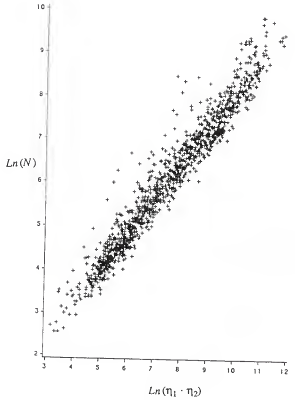
## Figure 5

$Ln(N)$ *vs.* $Ln(\eta_1 + \eta_2)$



# Figure 6

$Ln(N)$ *vs.* $Ln(\eta_1 \cdot \eta_2)$



The error term ( $\varepsilon^*$ ) handled in above two models (7.1) and (7.2) are multiplicative instead of additive. Therefore, the models of handling error terms in additive fashion are also considered as the form of:

$$N = \alpha \cdot (\eta_1 + \eta_2)^\beta + \varepsilon^+ \tag{8.1}$$

and,

$$N = \alpha \cdot (\eta_1 \cdot \eta_2)^\beta + \varepsilon^+ \tag{8.2}$$

The procedure of deriving the parameters in (8.1) and (8.2) is not the same as in (7.1) and (7.2). The parameters in (7.1) and (7.2) are obtained only by running simple linear regression on the transformed data. However, in the latter models, (8.1) and (8.2), the procedures of acquiring the parameter is by nonlinear least squares. Note the method in this procedure is modified Gauss-Newton method, and the data are processed by the procedure NLIN in SAS package [SAS-STAT 82].

For the convenience sake, all the models will be labelled by a particular symbol in the rest of this report. (See Table 1)

**The Description of Experiment**

The experiment is conducted starting from data collection, segmentation, then, followed by parameters development, length estimation, correlation of actual length versus estimated length, and relative error analysis. The results of the experiment are discussed in the following chapter.

**Table 1**

| Symbol | Model |
|---|---|
| $H$ | $\hat{N} = \eta_1 \cdot \log_2(\eta_1) + \eta_2 \cdot \log_2(\eta_2)$ |
| $F^+$ | $\hat{N} = c + \eta_2 \cdot \log_2(\eta_2)$ |
| $A^*$ | $\hat{N} = c \cdot \eta_2 \cdot \log_2(\eta_2)$ |
| $J$ | $\hat{N} = \log_2(\eta_1!) + \log_2(\eta_2!)$ |
| $L^+$ | $\hat{N} = \gamma \cdot (\eta_1 + \eta_2)^\beta$ |
| $L^*$ | $\hat{N} = \gamma \cdot (\eta_1 \cdot \eta_2)^\beta$ |
| $NL^+$ | $\hat{N} = \alpha \cdot (\eta_1 + \eta_2)^\beta$ |
| $NL^*$ | $\hat{N} = \alpha \cdot (\eta_1 \cdot \eta_2)^\beta$ |

Note: $L^+$ and $L^*$ are the models derived from simple linear regression of logorithmic transformation.

$NL^+$ and $NL^*$ are the models derived from nonlinear regression by least squares.

1) Data collection and partition:

There are three data sets used to investigate the estimation models:

- 799 UNIX system source codes (See Appendix A),
- 99 CMPSC 541 project programs written in C (See Appendix B), and
- 404 Pascal programs acquired from Mata-Toledo's dissertation [Mata 84] (See Appendix C).

The data set that includes all these three data sets is also observed.

In order to see the behavior of the errors on each model under various program length, the data are also partitioned into five parts by the size of the actual length.

i. Total (include all the observations)
ii. Actual length is less than or equal to 500.
iii. Actual length is between 501 and 1000.
iv. Actual length is between 1001 and 2000.
v. Actual length is between 2001 and 4000.
vi. Actual length is more than 4000.

2) Parameter Estimation:

According to the models described in this section, the parameters were estimated for all the models. The procedure of deriving the parameters of the model $L^+$, $L^*$, $NL^+$, and $NL^*$ are discussed in the previous section. There is no parameter needed in the model $H$ and $J$. The $c$ constant value in the model $F^+$ is obtained from averaging the $c$'s which are calculated in the individual observation. In the model $A^*$, the $c$ constant value is estimated by fitting a linear model without an intercept. Note that the parameters were estimated based upon the complete data sets, but not on the terspartitioned ones.

3) Estimated Length Acquisition and Relative Error Calculation:

The parameters were used in the model(s) to calculate the estimated length for each observation. After all the estimated length were obtained, then, the relative errors of each observation were calculated by the equation:

$$\text{Relative Error} = |\ \text{estimated length} - \text{actual length}\ | \ / \ \text{actual length}$$

4) Correlation Coefficients Comparisons:

The correlation coefficient is a measure of the linear relationship between two variables. These coefficients are calculated in order to examine the linear relationship between the estimated length and actual length for all the models in various data sets.

5) Mean of Relative Error:

The correlation coefficient estimates the degree of the closenesss of linear relationship between two variables. In these variables (estimated and actual length), the relative error is also important. However, the correlation coefficient does not provide information of the closeness of two variables. Therefore, the relative errors are also used. The mean of the relative error according to the combination of models and the partitioned data are computed. These values represent the accuracy of the estimation of each model. The number of over-estimated and under-estimated are also determined by comparing the estimated length and actual length.

# RESULTS AND DISCUSSIONS

The results of the experiment described in the previous chapter are presented in tabular form. Table 2.1, 2.2, 2.3 and 2.4 present the parameters developed based upon various data sets, such as Total, UNIX, Pascal and CMPSC 541. Table 3 shows the correlation coefficients between the estimated length and actual length under particular combination of model and data sets. The means of relative error are presented in the table 3. Concerning the various range of actual length data, the rest of the tables indicate the mean of relative error and count of over- and under- estimated of various range of actual program length. These tables are named as the form of X-Y, where X represents the name of the data set, and Y the model name. The accuracy of the model is defined as small MRE and balance of the counts of overestimating and underestimating of actual length.

### Correlation Coefficients of Actual length vs. Estimated Length

On observing the table 3, it can be seen that all the estimated values are highly correlated with the actual length in the various data sets. The correlation coefficients in most of the others are higher than 0.9 (Except for the data set of CMPSC 541). Roughly speaking, the models proposed in this report have the coefficients values a little bit higher than the others. (Except $NL^*$ in total set, $L^*$ in Pascal set, and $L^+$ in CMPSC 541 set). High correlation means that the two variables are likely to have a linear relationship. But, high correlation does not imply that the $N$ is equal or close to $\hat{N}$. In order to examine more detail of the estimation models, the term of relative error is employed,

**Table 2.1**

**Parameters Estimation ( Total observations )**

( Number of Observations = 1302 )

| Model | Parameter | S.E. |
|---|---|---|
| $H$ | None | |
| $F^+$ | $c$ = 526.3166 | 30.2102 |
| $A^*$ | $c$ = 1.52496 | 0.017142 |
| $J$ | None | |
| $L^+$ | $\gamma = -0.797189$ | 0.046869 |
| | $\beta = 1.523624$ | 0.010185 |
| $L^*$ | $\gamma = 0.060337$ | 0.039726 |
| | $\beta = 0.801774$ | 0.005156 |
| $NL^+$ | $\alpha = 2.111956$ | 0.189451 |
| | $\beta = 1.261011$ | 0.014111 |
| $NL^*$ | $\alpha = 1.085501$ | 0.165480 |
| | $\beta = 0.808215$ | 0.014440 |

Note:S.E. is the standard error of the estimation of corresponding parameter.

**Table 2.2**

**Parameters Estimation ( UNIX source programs )**

( Number of Observations = 799 )

| Estimator | Parameters | S.E. |
|---|---|---|
| $H$ | None | |
| $F^+$ | $c$ = 710.9088 | 38.4683 |
| $A^*$ | $c$ = 1.876848 | 0.023487 |
| $J$ | None | |
| $L^+$ | $\gamma = -0.938968$ | 0.063487 |
| | $\beta = 1.557691$ | 0.013077 |
| $L^*$ | $\gamma = 0.169702$ | 0.052626 |
| | $\beta = 0.785363$ | 0.006832 |
| $NL^+$ | $\alpha = 1.172288$ | 0.150518 |
| | $\beta = 1.368080$ | 0.021244 |
| $NL^*$ | $\alpha = 1.5572644$ | 0.185082 |
| | $\beta = 0.760995$ | 0.011297 |

Note:S.E. is the standard error of the estimation of corresponding parameter.

**Table 2.3**

**Parameters Estimation ( Pascal programs )**

( Number of Observations = 404 )

| Model | Parameter | S.E. |
|---|---|---|
| $H$ | None | |
| $F^{+}$ | $c$ = 174.2780 | 52.0660 |
| $A^{*}$ | $c$ = 1.28740 | 0.020870 |
| $J$ | None | |
| $L^{+}$ | $\gamma = -0.541532$ | 0.060009 |
| | $\beta = 1.426176$ | 0.014427 |
| $L^{*}$ | $\gamma = -0.217718$ | 0.060525 |
| | $\beta = 0.838510$ | 0.009033 |
| $NL^{+}$ | $\alpha = 1.1170146$ | 0.248648 |
| | $\beta = 1.344902$ | 0.032790 |
| $NL^{*}$ | $\alpha = 0.062634$ | 0.018846 |
| | $\beta = 1.120711$ | 0.028079 |

Note:S.E. is the standard error of the estimation of corresponding parameter.

**Table 2.4**

**Parameters Estimation ( CMPSC 541 programs )**

( Number of Observations = 99 )

| Model | Parameter | S.E. |
|---|---|---|
| $H$ | None | |
| $F^+$ | $c$ = 473.1435 | 92.7675 |
| $A^*$ | $c$ = 2.490146 | 0.135502 |
| $J$ | None | |
| $L^+$ | $\gamma = -1.051215$ | 0.333824 |
| | $\beta = 1.677929$ | 0.079486 |
| $L^*$ | $\gamma = 0.034634$ | 0.287546 |
| | $\beta = 0.856614$ | 0.041240 |
| $NL^+$ | $\alpha = 2.7808183$ | 1.066685 |
| | $\beta = 1.287783$ | 0.068851 |
| $NL^*$ | $\alpha = 1.808370$ | 0.713664 |
| | $\beta = 0.811828$ | 0.041813 |

Note:S.E. is the standard error of the estimation of corresponding parameter.

**Table 3**

**Correlation Coefficients of $N$ vs. $\hat{N}$**

| Model | Total (1302) | UNIX (799) | Pascal (404) | CMPSC 541 (99) |
|---|---|---|---|---|
| $H$ | 0.92994 | 0.92880 | 0.94333 | 0.86263 |
| $F^+$ | 0.90285 | 0.90523 | 0.94455 | 0.83653 |
| $A^*$ | 0.90285 | 0.90523 | 0.94455 | 0.83653 |
| $J$ | 0.92916 | 0.92892 | 0.94432 | 0.85968 |
| $L^+$ | 0.91734 | 0.92728 | 0.94341 | 0.82119 |
| $L^*$ | 0.88873 | 0.94112 | 0.93455 | 0.88478 |
| $NL^+$ | 0.93191 | 0.93225 | 0.94466 | 0.85978 |
| $NL^*$ | 0.88872 | 0.94129 | 0.94928 | 0.88684 |

and discussed in the following secitons.

**Mean Of Relative Error (MRE) Analysis:**

The values of the mean of relative errors are listed in Table 4; the values are obtained by observing the full range of the various data sets. The value in the parenthesis of each box presents the rank of the models by the value of MRE for the particular data set. For a more detail investigation of the behavior of the MRE, each data set was partitioned into five parts according to the actual length of the programs. The MRE and the counts of over- and under- estimated are illustrated in the tables from page of 36 to 43. These tables are arranged according to the combination of the model and the data set which is observed. In each table, the first column shows the range of the actual length, the second column indicates the number of the observations, the MRE, and the counts of over- and under- estimated observation are listed in the column 3, 4 and 5 respectively.

The table 4 shows the models of $L^+$ and $L^*$ have smaller MRE than most other models. Model $F^+$ has largest MRE in the listed eight models. If we sum the rank ( values in the parentheses ) for each model, then the superiority rank of the models can simply drawn by this sum. Models $L^+$ and $L^*$ are ranked first (6), being followed by $J$ (16), $NL^*$ (18), $A^*$ (18), $H$ (22), $NL^+$ (26), and $F^+$ (32). It is not reasonable to say that these results are final; the behavior of MRE in different range of actual length, the counts of overestimated and under estimated also need to be investigated.

## Table 4

## Mean Relative Error Comparisons

| Model | Total | UNIX | Pascal | CMPSC 541 |
|---|---|---|---|---|
| $H$ | 0.43321 (6) | 0.36637 (6) | 0.58857 (7) | 0.33869 (3) |
| $F^+$ | 3.11230 (8) | 2.69839 (8) | 1.79549 (8) | 1.68634 (8) |
| $A^*$ | 0.36258 (5) | 0.32643 (5) | 0.35115 (4) | 0.35556 (4) |
| $J$ | 0.29666 (4) | 0.30660 (4) | 0.26254 (3) | 0.35571 (5) |
| $L^+$ | 0.27382 (2) | 0.24802 (2) | 0.24323 (1) | 0.32592 (1) |
| $L^*$ | 0.25430 (1) | 0.23204 (1) | 0.25147 (2) | 0.32858 (2) |
| $NL^+$ | 0.66637 (7) | 0.40005 (7) | 0.50958 (5) | 0.85043 (7) |
| $NL^*$ | 0.28892 (3) | 0.27106 (3) | 0.57086 (6) | 0.53149 (6) |

The characteristics of each model's estimation are discussed in the following sections:

$H$ ( $\hat{N} = \eta_1 \cdot \log_2(\eta_1) + \eta_2 \cdot \log_2(\eta_2)$ ):

The bias of Halstead's model was inspected (page 36); it tends to overestimat in the small programs but underestimate in the large program size. In the programs of length less than 500, $H$ has about 90 percent of the time overestimated the actual length; in contrast lengths greater than 4000, $H$ underestimated the actual length more than 92 percents of the time (page 36). In the sets of C541 programs, it always overetimated when the actual length is greater than 500, and MRE apparantly increases when $N$ goes up. The range of actual length between 500 and 1000 seems more suitable for the Halstead's model, that is of more balance of over and under estimation and smaller MRE.

$F^+$ ( $\hat{N} = c + \eta_2 \cdot \log_2(\eta_2)$ ):

Because of high variation of the constant $c$ in the Fitsos's model, the accuracy of this model has been questioned. The coefficient of variations of $c$ are 207.1156, 152.9544, 600.4850 and 195.0836 in the data sets of Total, UNIX, Pascal and C541 respectively. The results of MRE also show the obvious bias of Fitsos's model, not only high MRE but also seriously overestimating the small size of the program (page 37).

$A^*$ ( $\hat{N} = c \cdot \eta_2 \cdot \log_2(\eta_2)$ ):

The model of $A^*$ behaves more consistently than $H$ or $A^*$. There is no obvious trend of MRE and of the unbalance of counts of over/under estimation appearing when the program length changed. In the Total and UNIX data sets (see page 38), it becomes more accurate when the actual length grows, the results also agrees with Fitsos's assumptions that the number of operator becomes constant when the program length increases. It has a very high MRE and an unbalance of over/under estimating counts in the Pascal data set of mid-size range of the $N$.

$J$ ( $\hat{N} = \log_2(\eta_1!) + \log_2(\eta_2!)$ ):

This model has been investigated by Jensen [Jensen 85], who used the data of the length less than 400 in the average. According to this range of the length, author found agreement with Jensen's results, that model $J$ is a quite good model when the actual program length less than 500. However, when the actual length greater than 500 being observed, the trend of the MRE occurs (page 39). This trend shows the model $J$ tends to under estimate the actual length when it increases. This phenomena appears in all four data sets, for example in the UNIX data set, the MRE in the size less than 500 is 0.24601, and percentage of underestimating counts is 43, but in the size greater than 4000, MRE become 0.47555 and percentage of underestimating counts becomes 100. It has almost 100 percent of time underestimated the actual length when $N$ was greater than 4000.

$L^+$ ( $\hat{N} = \gamma \cdot (\eta_1 + \eta_2)^\beta$ ):

The model $L^+$ has very low MRE and balance of over/under estimating counts in all four data sets (page 40). This is dure to the model being derived from a least square approach. There is a little bit higher MRE in Pascal data set in the range of 500 to 2000 ( MRE is about 0.48 ), but it is still lower than that of most of other models.

$L^*$ ( $\hat{N} = \gamma \cdot (\eta_1 \cdot \eta_2)^\beta$ ):

In the viewpoint of accuracy, models $L^*$ and $L^+$ are very similar (page 41). It has low MRE and balance of over/under estimating counts. These two models, $L^+$ and $L^*$, shows the lowest MRE of eight models in current study.

$NL^+$ ( $\hat{N} = \alpha \cdot (\eta_1 + \eta_2)^\beta$ ):

The model $NL^+$ tends to overestimates when actual length is small. It has highest MRE and percentage of overestimating counts in the length smaller than 500, and become more accuracy when the size increases (page 42). (except the size greater than 4000 in the C541 data set).

$NL^*$ ( $\hat{N} = \alpha \cdot (\eta_1 \cdot \eta_2)^\beta$ ):

This model has a low MRE, but the count of over/under estimating seems to be language dependent, in the programs of short length. For instance, the programs of the length less than 500, there are more than 78 percents of the time overestimate the length in UNIX data set (page 43), and 90 percents in C541 data set. In contrast to this result, there are 99 percents of the time underestimate the length in the Pascal set.

**Summary**

From the above analysis, the models of $L^+$ and $L^*$ are suggested as the program length estimation. Not only do they have lower MRE but also the balance of over/under estimating counts is good. The model $H$ tends to overestimate the small program, and underestimate the large programs. Model $F^+$ has very high MRE's so that the model is not suggested for estimating. Model $A^*$ is good when the actual length is large. Model $J$ has serious bias dealing with the large programs, since the trend of MRE is existent; however, in the small data set it provides very impressive outcomes. For its simple structure and being parameter free, model $J$ is suggested when the program length is not very large. Model $NL^+$ has higher MRE, and $NL^*$ shows the results language dependent in small size programs, besides, the parametric values development in these two models is very time consuming, so $NL^+$ and $NL^*$ are not recommended for the length estimation.

From the view point of correlation coefficients, these eight models provided estimated length highly correlated with actual length. Nevertheless, some more justifications are required for most of them so that the model can function much better in estimation.

**Table Total - H**

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 1302 | 0.43321 | 803 | 498 |
| N < 501 | 702 | 0.55397 | 618 | 83 |
| 500 < N < 1001 | 165 | 0.25986 | 81 | 84 |
| 1000 < N < 2001 | 186 | 0.30685 | 77 | 109 |
| 2000 < N < 4001 | 148 | 0.26298 | 24 | 124 |
| N > 4000 | 101 | 0.35923 | 3 | 98 |

**Table UNIX - H**

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 799 | 0.36637 | 413 | 385 |
| N < 501 | 315 | 0.54309 | 280 | 34 |
| 500 < N < 1001 | 135 | 0.21209 | 72 | 63 |
| 1000 < N < 2001 | 149 | 0.22852 | 48 | 101 |
| 2000 < N < 4001 | 131 | 0.25918 | 12 | 119 |
| N > 4000 | 69 | 0.36267 | 1 | 68 |

**Table Pascal - H**

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 404 | 0.58857 | 348 | 56 |
| N < 501 | 318 | 0.62077 | 296 | 22 |
| 500 < N < 1001 | 14 | 0.61437 | 9 | 5 |
| 1000 < N < 2001 | 32 | 0.65151 | 29 | 3 |
| 2000 < N < 4001 | 13 | 0.23541 | 12 | 1 |
| N > 4000 | 27 | 0.29136 | 2 | 25 |

**Table C541 - H**

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 99 | 0.33869 | 42 | 57 |
| N < 501 | 69 | 0.29584 | 42 | 27 |
| 500 < N < 1001 | 16 | 0.35269 | 0 | 16 |
| 1000 < N < 2001 | 5 | 0.43507 | 0 | 5 |
| 2000 < N < 4001 | 4 | 0.47689 | 0 | 4 |
| N > 4000 | 5 | 0.67826 | 0 | 5 |

**Table Total - $F^+$**

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 1302 | 3.11230 | 965 | 337 |
| N < 501 | 702 | 5.47453 | 702 | 0 |
| 500 < N < 1001 | 165 | 0.37653 | 151 | 14 |
| 1000 < N < 2001 | 186 | 0.31914 | 88 | 98 |
| 2000 < N < 4001 | 148 | 0.31671 | 19 | 129 |
| N > 4000 | 101 | 0.40336 | 5 | 96 |

**Table UNIX - $F^+$**

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 799 | 2.69839 | 554 | 245 |
| N < 501 | 315 | 6.28746 | 315 | 0 |
| 500 < N < 1001 | 135 | 0.60655 | 135 | 0 |
| 1000 < N < 2001 | 149 | 0.21640 | 91 | 58 |
| 2000 < N < 4001 | 131 | 0.25583 | 13 | 118 |
| N > 4000 | 69 | 0.40324 | 0 | 69 |

**Table Pascal - $F^+$**

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 404 | 1.79549 | 365 | 39 |
| N < 501 | 318 | 2.16064 | 315 | 3 |
| 500 < N < 1001 | 14 | 0.57567 | 9 | 5 |
| 1000 < N < 2001 | 32 | 0.60740 | 28 | 4 |
| 2000 < N < 4001 | 13 | 0.19744 | 11 | 2 |
| N > 4000 | 27 | 0.30485 | 2 | 25 |

**Table C541 - $F^+$**

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 99 | 1.68634 | 76 | 23 |
| N < 501 | 69 | 2.29610 | 69 | 0 |
| 500 < N < 1001 | 16 | 0.12844 | 7 | 9 |
| 1000 < N < 2001 | 5 | 0.27222 | 0 | 5 |
| 2000 < N < 4001 | 4 | 0.45518 | 0 | 4 |
| N > 4000 | 5 | 0.65602 | 0 | 5 |

**Table Total - A***

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 1302 | 0.36258 | 454 | 848 |
| N < 501 | 702 | 0.33771 | 292 | 410 |
| 500 < N < 1001 | 165 | 0.41478 | 44 | 121 |
| 1000 < N < 2001 | 186 | 0.47152 | 66 | 120 |
| 2000 < N < 4001 | 148 | 0.32260 | 33 | 115 |
| N > 4000 | 101 | 0.30813 | 19 | 82 |

**Table UNIX - A***

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 799 | 0.32643 | 355 | 444 |
| N < 501 | 315 | 0.37425 | 155 | 160 |
| 500 < N < 1001 | 135 | 0.37337 | 62 | 73 |
| 1000 < N < 2001 | 149 | 0.32088 | 72 | 77 |
| 2000 < N < 4001 | 131 | 0.23192 | 51 | 80 |
| N > 4000 | 69 | 0.20768 | 15 | 54 |

**Table Pascal - A***

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 404 | 0.35115 | 187 | 217 |
| N < 501 | 318 | 0.28707 | 129 | 189 |
| 500 < N < 1001 | 14 | 0.75631 | 9 | 5 |
| 1000 < N < 2001 | 32 | 0.87499 | 29 | 3 |
| 2000 < N < 4001 | 13 | 0.42121 | 12 | 1 |
| N > 4000 | 27 | 0.24126 | 8 | 19 |

**Table C541 - A***

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 99 | 0.35556 | 69 | 30 |
| N < 501 | 69 | 0.38227 | 56 | 13 |
| 500 < N < 1001 | 16 | 0.29103 | 9 | 7 |
| 1000 < N < 2001 | 5 | 0.17190 | 2 | 3 |
| 2000 < N < 4001 | 4 | 0.16545 | 1 | 3 |
| N > 4000 | 5 | 0.52914 | 1 | 4 |

**Table Total - J**

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 1302 | 0.29666 | 468 | 834 |
| N < 501 | 702 | 0.24204 | 396 | 306 |
| 500 < N < 1001 | 165 | 0.30509 | 27 | 138 |
| 1000 < N < 2001 | 186 | 0.34242 | 37 | 149 |
| 2000 < N < 4001 | 148 | 0.37448 | 7 | 141 |
| N > 4000 | 101 | 0.46428 | 1 | 100 |

**Table UNIX - J**

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 799 | 0.30660 | 212 | 587 |
| N < 501 | 315 | 0.24601 | 178 | 137 |
| 500 < N < 1001 | 135 | 0.26569 | 19 | 116 |
| 1000 < N < 2001 | 149 | 0.32052 | 13 | 136 |
| 2000 < N < 4001 | 131 | 0.38967 | 2 | 129 |
| N > 4000 | 69 | 0.47555 | 0 | 69 |

**Table Pascal - J**

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 404 | 0.26254 | 245 | 159 |
| N < 501 | 318 | 0.23352 | 207 | 111 |
| 500 < N < 1001 | 14 | 0.44932 | 8 | 6 |
| 1000 < N < 2001 | 32 | 0.40956 | 24 | 8 |
| 2000 < N < 4001 | 13 | 0.15619 | 5 | 8 |
| N > 4000 | 27 | 0.38438 | 1 | 26 |

**Table C541 - J**

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 99 | 0.35571 | 11 | 88 |
| N < 501 | 69 | 0.26318 | 11 | 58 |
| 500 < N < 1001 | 16 | 0.51137 | 0 | 16 |
| 1000 < N < 2001 | 5 | 0.56548 | 0 | 5 |
| 2000 < N < 4001 | 4 | 0.58647 | 0 | 4 |
| N > 4000 | 5 | 0.74022 | 0 | 5 |

**Table Total - $L^+$**

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 1302 | 0.27382 | 684 | 618 |
| N < 501 | 702 | 0.24372 | 434 | 268 |
| 500 < N < 1001 | 165 | 0.30436 | 65 | 100 |
| 1000 < N < 2001 | 186 | 0.38194 | 90 | 96 |
| 2000 < N < 4001 | 148 | 0.26436 | 60 | 88 |
| N > 4000 | 101 | 0.24792 | 35 | 66 |

**Table UNIX - $L^+$**

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 799 | 0.24802 | 424 | 375 |
| N < 501 | 315 | 0.24843 | 201 | 114 |
| 500 < N < 1001 | 135 | 0.25507 | 57 | 78 |
| 1000 < N < 2001 | 149 | 0.27545 | 68 | 81 |
| 2000 < N < 4001 | 131 | 0.23769 | 64 | 67 |
| N > 4000 | 69 | 0.19275 | 34 | 35 |

**Table Pascal - $L^+$**

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 404 | 0.24323 | 217 | 187 |
| N < 501 | 318 | 0.20628 | 168 | 150 |
| 500 < N < 1001 | 14 | 0.47496 | 8 | 6 |
| 1000 < N < 2001 | 32 | 0.48804 | 25 | 7 |
| 2000 < N < 4001 | 13 | 0.18844 | 9 | 4 |
| N > 4000 | 27 | 0.29449 | 7 | 20 |

**Table C541 - $L^+$**

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 99 | 0.32592 | 61 | 38 |
| N < 501 | 69 | 0.33915 | 48 | 21 |
| 500 < N < 1001 | 16 | 0.28732 | 8 | 8 |
| 1000 < N < 2001 | 5 | 0.15942 | 2 | 3 |
| 2000 < N < 4001 | 4 | 0.14576 | 2 | 2 |
| N > 4000 | 5 | 0.57761 | 1 | 4 |

**Table Total - L***

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 1302 | 0.25430 | 745 | 557 |
| N < 501 | 702 | 0.23702 | 436 | 266 |
| 500 < N < 1001 | 165 | 0.28552 | 84 | 81 |
| 1000 < N < 2001 | 186 | 0.28418 | 113 | 73 |
| 2000 < N < 4001 | 148 | 0.24847 | 79 | 69 |
| N > 4000 | 101 | 0.27696 | 33 | 68 |

**Table UNIX - L***

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 799 | 0.23204 | 449 | 350 |
| N < 501 | 315 | 0.24184 | 212 | 103 |
| 500 < N < 1001 | 135 | 0.23256 | 69 | 66 |
| 1000 < N < 2001 | 149 | 0.23361 | 80 | 69 |
| 2000 < N < 4001 | 131 | 0.23568 | 61 | 70 |
| N > 4000 | 69 | 0.17600 | 27 | 42 |

**Table Pascal - L***

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 404 | 0.25147 | 221 | 183 |
| N < 501 | 318 | 0.20338 | 174 | 144 |
| 500 < N < 1001 | 14 | 0.58552 | 9 | 5 |
| 1000 < N < 2001 | 32 | 0.48848 | 27 | 5 |
| 2000 < N < 4001 | 13 | 0.19022 | 9 | 4 |
| N > 4000 | 27 | 0.39326 | 2 | 25 |

**Table C541 - L***

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 99 | 0.32858 | 60 | 39 |
| N < 501 | 69 | 0.35174 | 48 | 21 |
| 500 < N < 1001 | 16 | 0.29335 | 8 | 8 |
| 1000 < N < 2001 | 5 | 0.16039 | 2 | 3 |
| 2000 < N < 4001 | 4 | 0.12378 | 1 | 3 |
| N > 4000 | 5 | 0.45380 | 1 | 4 |

**Table Total - NL⁺**

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 1302 | 0.66637 | 1017 | 285 |
| N < 501 | 702 | 0.95911 | 679 | 23 |
| 500 < N < 1001 | 165 | 0.37928 | 121 | 44 |
| 1000 < N < 2001 | 186 | 0.40033 | 127 | 59 |
| 2000 < N < 4001 | 148 | 0.22919 | 71 | 77 |
| N > 4000 | 101 | 0.23128 | 19 | 82 |

**Table UNIX - NL⁺**

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 799 | 0.40005 | 561 | 238 |
| N < 501 | 315 | 0.62559 | 292 | 23 |
| 500 < N < 1001 | 135 | 0.30143 | 99 | 36 |
| 1000 < N < 2001 | 149 | 0.27761 | 89 | 60 |
| 2000 < N < 4001 | 131 | 0.21496 | 64 | 67 |
| N > 4000 | 69 | 0.17921 | 17 | 52 |

**Table Pascal - NL⁺**

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 404 | 0.50958 | 344 | 60 |
| N < 501 | 318 | 0.51024 | 286 | 32 |
| 500 < N < 1001 | 14 | 0.63585 | 9 | 5 |
| 1000 < N < 2001 | 32 | 0.72585 | 29 | 3 |
| 2000 < N < 4001 | 13 | 0.34342 | 12 | 1 |
| N > 4000 | 27 | 0.26000 | 8 | 19 |

**Table C541 - NL⁺**

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 99 | 0.85043 | 84 | 15 |
| N < 501 | 69 | 1.08063 | 67 | 2 |
| 500 < N < 1001 | 16 | 0.35793 | 11 | 5 |
| 1000 < N < 2001 | 5 | 0.19924 | 3 | 2 |
| 2000 < N < 4001 | 4 | 0.13099 | 2 | 2 |
| N > 4000 | 5 | 0.47635 | 1 | 4 |

Table Total - NL*

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 1302 | 0.28892 | 844 | 458 |
| N < 501 | 702 | 0.27265 | 486 | 216 |
| 500 < N < 1001 | 165 | 0.31585 | 96 | 69 |
| 1000 < N < 2001 | 186 | 0.33230 | 125 | 61 |
| 2000 < N < 4001 | 148 | 0.28667 | 93 | 55 |
| N > 4000 | 101 | 0.28150 | 44 | 57 |

Table UNIX - NL*

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 799 | 0.27106 | 512 | 287 |
| N < 501 | 315 | 0.32693 | 247 | 68 |
| 500 < N < 1001 | 135 | 0.25292 | 82 | 53 |
| 1000 < N < 2001 | 149 | 0.24686 | 89 | 60 |
| 2000 < N < 4001 | 131 | 0.23482 | 66 | 65 |
| N > 4000 | 69 | 0.17260 | 28 | 41 |

Table Pascal - NL*

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 404 | 0.57086 | 55 | 349 |
| N < 501 | 318 | 0.59838 | 1 | 317 |
| 500 < N < 1001 | 14 | 0.59835 | 8 | 6 |
| 1000 < N < 2001 | 32 | 0.62654 | 26 | 6 |
| 2000 < N < 4001 | 13 | 0.42233 | 11 | 2 |
| N > 4000 | 27 | 0.23807 | 9 | 18 |

Table C541 - NL*

| Actual Length | Obs. | MRE | Over | Under |
|---|---|---|---|---|
| N > 0 | 99 | 0.53149 | 80 | 19 |
| N < 501 | 69 | 0.61816 | 62 | 7 |
| 500 < N < 1001 | 16 | 0.38630 | 10 | 6 |
| 1000 < N < 2001 | 5 | 0.20433 | 4 | 1 |
| 2000 < N < 4001 | 4 | 0.16734 | 3 | 1 |
| N > 4000 | 5 | 0.41846 | 1 | 4 |

# CONCLUSIONS AND FUTURE WORK

The study attempted to illustrate appropriate statistical methods for length measuring, and for adjusting measures for the effect of size. The estimation can not be a case independent work, it must be based upon the results obtained in the past. In [DeMarco 82, pp.6-7], DeMarco asserted the principle of the measurement:

> " Measurement is always a recording of past effects. The uses we will want to make of our measurement nearly always involve some predictive quantification of future effect. ... the estimating function is based rigorously on statistics collected from past activities."

In the real world, the various specifications, programming tools, even the personnel involved in the project, will all be factors that influence the parametric value in the model. The key point is that a model can efficiently and accurately utilize the past record and then develop more reliable parameters to estimate the software length.

Four models were proposed based upon the idea of linear regression modeling, and the data sets were transformed in order to meet the requirement of statistics features. Including the other four models proposed by various articles, [Halstead 72], [Fitsos 80], [Albrecht 83] and [Jensen 85], eight models were analyzed and compared. Not only were correlation analysis done between the estimated and the actual length, but also the mean of relative error and counts of over/under estimating techniques were employed in the comparison tasks. The results of the models $L^+$ and $L^*$, proposed by the author, were more precise in estimating the length than the other models. They provided smaller MRE and balanced the over/under estimating counts.

**Future Work**

In this report, the logarithmic transformation was employed in order to transform the data set. There are other ways to transform the data, but that analysis will be left for the future work. There still needs more attention to analyzing the trend of the MRE behavior in some models, such as $H$, $J$, and $A^*$. Those provide moderate MRE; however, it shows a trend of bias depending upon whether the actual program length increases or decreases. The modeling methods introduced in this report could also be useful in the estimation of other program characteristics. This also deserves more work in the future.

# REFERENCE


[Albrecht 83] A. J. Albrecht, and J. E. Gaffney, "Software Function, Source Lines Of Code and Developement Effort Prediction: Software Sciencr Validation", *IEEE Trans. on Software Engineering,* SE-9, no.1, January 1983, pp.639-648.

[Baker 79] A. L. Baker and S. H. Zweben, "The Use of Software Science in Evaluating Modularity Concepts," *IEEE Trans. Software Engineering*, vol. SE-5, no. 2, March 1979, pp. 110-120.

[Basili 80] V. R. Basili, T. Phillips, "Evaluating and Comparing Software Metrics in the Software Engineering Laboratory", *ACM SIG-METRICS*, (1981 ACM Workshop/Symposium on Measurement and Evaluation of Software Quality), vol. 10, pp. 95-106, March 1980.

[Basili 83] V. R. Basili, R. W. Selby, and T. Y. Phillips, "Metric Analysis and Data Validation," *IEEE Trans. Software Engineering*, vol. SE-9, no.6, March 1983, pp. 652-663.

[Boehm 76] B. W. Boehm, "Software Engineering". *IEEE Trans. on Computers,* vol. C- 25, no. 12, Dec. 1976, pp. 1226-1241.

[Boehm 84] B. W. Boehm, "Software Engineering Economics," *IEEE Trans. on Software Engineering,* SE-10, no. 1, January 1984, pp. 4-21.

[Britcher 85] R. N. Britcher, and F. E. Gaffney, "Reliable Size Estimates For Software Systems Decomposed As State Machines," *Proceedings of COMPSAC 85,* October 1985, pp.104-106.

[Bulut 74] N. Bulut, M. H. Halstead, and R. Bayer, "Experimental Validation Of a Structural Property Of FORTRAN Algrothms", *Proceedings of the ACM National Conference,* 1974, pp.207-211.

[Chris 81] K. Christensen, G. P. Fitsos, and C. P. Smith, "A Perspective on Software Science". *IBM System Journal* vol. 20, no.4, April 1981, pp. 372-387.

[Comer 79] D. Comer and M.H. Halstead, "A Simple Experiment in Top-Down Design", *IEEE Trans. on Software Engineering,* SE-5, no. 3, March 1979, pp. 105-109.

[Cook 82] M. L. Cook, "Software Metrics: An Introduction and Annotated Bibliography", *Software Engineering Notes,* vol. 7, no. 2, April 1982, pp. 41-60.

[Coulter 83] N. S. Coulter, "Software Science and Congnitive Psychology," *IEEE Trans. Software Engineering,* vol. SE-9, no. 6, March 1983, pp. 105-109.

[Crawford 85] S. G. Crawford, A. A. McIntosh, and D. Pregibon, "An Analysis Of Static Metrics and Faults In C Software," *The Journal Of System And Software,* vol 5, May 1985, pp. 37-48.

[Curtis 79] B. Curtis, S. B. Sheppard, and P. Milliman, "Third Time Charm: Stronger Prediction of Programmer Performance by Software Complexity Metrics", *Proceedings of the 4th International Conference on Software Engineering,* 1979, pp. 356-360.

[Curtis 80] B. Curtis, "Measurement and Experimentation in Software Engineering", *Proceedings of the IEEE*, vol. 68, no. 9, September 1980, pp. 1144-1157.

[DeMacro 82] T. DeMacro,"*Controlling Software Projects*", New York: Yourdon Inc., 1982

[Dijkstra 72] E. W. Dijkstra,"Notes On Structure Programming", *Structured Programming*, A.P.I.C. Studies in Data Processing, No. 8, Academic press, 1972.

[Elshoff 76] J. L. Elshoff, "Measuring Commercial PL/I Programs Using Halstead's Criteria", *ACM SIGPLAN Notices*, vol.7, no.5, May 1976.

[Elshoff 78] J. L. Elshoff, "An Investigation Into the Effects of the Counting Method Used On Software Science Measurement," *ACM SIGPLAN NOTICES.* vol. 13, no. 2, Feb. 1978.

[Feuer 79] A. R. Feuer and E. B. Fowlkes, "Some Results From an Empirical Study of Computer Software," *4th International Conference on Software Engineering,* September 1979, pp. 351-355.

[Fitsos 80] G. P. Fitsos, "Vocabulary effects of Software Science," *Proceedings of COMPSAC 80*, October 1980, pp. 751-756.

[Fitz 78] A. Fitzsimmons and T. Love, "A Review and Evaluation of Software Science," *ACM Computing Survey.* vol. 10, no. 1, March 1978, pp. 3-18.

[Gustafson 81] D. A. Gustafson, and R. R. Meals, "An Experiment In the Implementation And Application Of Halstead's and McCabe's Measures and Complexity," *Proceedings of Software Engineering Standards 81,* August 1981, pp. 45-50.

[Halstead 72] M. H. Halstead, "Natural Laws Controlling Algorithm Structure", *ACM SIGPLAN Notices*, vol.3, no. 2 Feb. 1972.

[Halstead 77] M. H. Halstead, "*Elements of Software Science*," New York: Elsevier, North Holland, 1977.

[Hamer 82] P. G. Hammer, G. D. Frewin, "M. H. Halstead's Software Science - A Critical examination", *Proceedings of the 6th International Conference on Software Engineering*, Tokyo, September 1982, pp. 197-206.

[Itakura 82] M. Itakura, and A. Takayanagi, "A Model for Estimating Program Size and Its Evaluation," *Proceedings of the Sixth International Conference on Software Engineering*, September 1982, pp. 104-109.

[Ivan 87] I. Ivan, R. Arhire, and M. Macesanu, "Program Complexity: Comparative Analysis, Hierarchy, Classification", *ACM SIGPLAN NOTICES*, vol. 22, no. 4, April 1987, pp. 94-102.

[Iyengar 82] S. S. Iyengar, N. Parameswaran, and J. W. Fuller, "A measurement of Logical Complexity of Programs", *Journal of Computer Language*, vol. 7, 1982, pp. 147-160.

[Jensen 85] H. A. Jensen, K. Vairavan, "An experimental study of Software Metrics for real-time Software", *IEEE Trans. on Software Engineering*, vol. SE-11, no. 2, February 1985, pp. 231-234.

[Konstam 85] A. H. Konstam, and D. E. Wood, "Software Science Applied to APL," *IEEE Trans. on Software Engineering*, vol.SE-11, no. 10, Oct. 1985, pp. 994-1000.

[Lassez 81] J. Lassez, D. Van Der Knijff, and J. Shepherd, "A Critical Examination of Software Science," *Journal System Software*, vol. 2, Dec 1981, pp. 105-112.

[Leslie 87] J. W. Leslie, and S. Badlani, "Software Complexity Asswssment: An Introduction And Annotated Bibliography", *ACM SIGSOFT Software Engineering Notes*, vol. 12, no. 4, Oct. 1987, pp. 52-71.

[Levitin 85] A. V. Levitin, "On Predicting Program Size by Program Vocabulary," *Proceedings of COMPSAC 85*, October 1985, pp.98-103.

[Levitin 86] A. V. Levitin, "How To Measure Software Size and How Not To," *Proceedings of COMPSAC 86*, October 1986, pp. 314-318.

[Levitin 87] A. V. Levitin, "Investigating Predictability Of Program Size," *Proceedings of COMPSAC 87*, October 1987, pp. 231-235.

[Mata 84] R. A. Mata-Toledo, "*Factor Analysis Of Complexity Measures*," Ph. D. Dissertation, Department Of Computer And Information Sciences, Kansas State University, 1984.

[Mohanty 79] S. N. Mohanty, "Models And Measurements For Quality Assessment Of Software", *Computing Survey*, vol. 11, no.3, Oct. 1979, pp. 251-275.

[Moranda 81] P. Moranda, "Is Software Science hard?", *Computing Surveys*, vol. 10, December 1981, pp.503-504.

[Moranda 85] P. Moranda, Review of Shooman's "Software Engineering: Reliability, Development, and Management," *Computing Review*, vol. 26, no. 2, Feb. 1985, pp. 92-93.

[Ross 75] D. T. Ross et al., "Software Engineering: Process Principles, and Goals," *IEEE Computer* vol. 8, no. 5, 1975.

[SAS-STAT 82] SAS Institute Inc. "SAS User's Guide: Statistics", 1982 Edition. Cary, NC: SAS Institute Inc., 1982. pp 15-37.

[Shen 79a] V. Y. Shen, and H. E. Dunsmore, "Analyzing Cobol Programs via Software Science," *IEEE Trans. Software Engineering*, vol. SE-5, March 1979, pp 79-90.

[Shen 79b] V. Y. Shen, "The Relationship between Student grades and Software Science Parameters", *COMPSAC 79*, November 1979, pp. 783-787.

[Shen 83] V. Y. Shen, S. D. Conte, and H. E. Dunsmore, "Software Science Revisited: A Critical Analysis of the Theory and Its Empirical Support," *IEEE Trans. Software Engineering*, vol. SE-9, no. 3, March 1983, pp. 155-165.

[Shneid 86] B. Shneiderman, "Empirical Studies of Programmers: the territory, paths, and destinations", in *Empirical Studies of Programmers*. Soloway, E., and Iyengar, S. E.(eds), Ablex Publishing Corp, Norwood, New Jersey, June 1986, pp. 1-12.

[Shooman 83] M. L. Shooman, "Software Engineering: Reliability, Development, and Management," McGraw-Hill, Inc., New York 1983.

[Smith 80] C. P. Smith, "A Software Science Analysis Of Programming Size", *Proceedings ACM, National Computing Conference*, Oct. 1980, pp. 179-185.

[Walston 77] C. E. Walston and C. P. Felix, "A Method of Programming Measurement and Estimation", *IBM System Journal*, vol 16, no. 1, Jan. 1977.

[Woodfield 79] S. N. Woodfield, "An Experiment On Unit Increase In Problem Complexity," *IEEE Trans. on Software Engineering*, vol.SE-5, no.2, March 1979, pp. 76-79.

[Woodfield 81] S. N. Woodfield, V. Y. Shen, and H. E. Dunsmore, "A Study Of Several Metrics For Programming Effort", *Journal System Software*, vol.2, pp. 97-103, Dec. 1981.

[Zweben 79] S. H. Zweben, and K. Fung, "Exploring Software Science Relations in COBOL and APL", *Proceedings of the 3rd International Conference on Software Engineering*, Chicago IL, 1979, pp. 702-707.

# Appendix A

**Token Counts Of UNIX Source Programs**

| $\eta_1$ | $\eta_2$ | $N_1$ | $N_2$ | $N$ |
|---|---|---|---|---|
| 78 | 193 | 1513 | 860 | 2373 |
| 77 | 193 | 1517 | 865 | 2382 |
| 89 | 134 | 936 | 465 | 1401 |
| 73 | 194 | 1332 | 735 | 2067 |
| 48 | 47 | 323 | 168 | 491 |
| 78 | 111 | 686 | 327 | 1013 |
| 99 | 124 | 1178 | 500 | 1678 |
| 19 | 167 | 260 | 201 | 461 |
| 95 | 120 | 990 | 408 | 1398 |
| 39 | 30 | 195 | 78 | 273 |
| 68 | 86 | 469 | 206 | 675 |
| 77 | 172 | 941 | 498 | 1439 |
| 80 | 99 | 982 | 431 | 1413 |
| 53 | 58 | 290 | 131 | 421 |
| 99 | 199 | 984 | 477 | 1461 |
| 83 | 118 | 1030 | 473 | 1503 |
| 71 | 114 | 588 | 309 | 897 |
| 72 | 97 | 715 | 341 | 1056 |
| 67 | 87 | 921 | 405 | 1326 |
| 44 | 31 | 282 | 108 | 390 |
| 73 | 100 | 1856 | 962 | 2818 |
| 30 | 9 | 127 | 45 | 172 |
| 78 | 114 | 1147 | 611 | 1758 |
| 53 | 50 | 337 | 168 | 505 |
| 35 | 136 | 479 | 265 | 744 |
| 42 | 34 | 525 | 355 | 880 |
| 36 | 138 | 485 | 267 | 752 |
| 145 | 198 | 3177 | 1639 | 4816 |
| 28 | 179 | 327 | 206 | 533 |
| 72 | 83 | 1036 | 578 | 1614 |
| 38 | 108 | 1175 | 1068 | 2243 |
| 81 | 80 | 761 | 333 | 1094 |
| 155 | 313 | 4622 | 2174 | 6796 |
| 34 | 26 | 236 | 132 | 368 |
| 185 | 646 | 7838 | 4274 | 12112 |
| 56 | 41 | 346 | 171 | 517 |
| 58 | 85 | 585 | 342 | 927 |
| 30 | 30 | 113 | 52 | 165 |
| 70 | 70 | 679 | 352 | 1031 |
| 106 | 236 | 3930 | 2019 | 5949 |
| 97 | 161 | 1646 | 984 | 2630 |
| 95 | 204 | 1679 | 865 | 2544 |
| 17 | 132 | 599 | 263 | 862 |
| 61 | 65 | 744 | 479 | 1223 |
| 165 | 354 | 5389 | 3041 | 8430 |
| 151 | 278 | 3510 | 2012 | 5522 |
| 120 | 423 | 2951 | 1500 | 4451 |
| 144 | 292 | 4997 | 3163 | 8160 |
| 25 | 89 | 181 | 105 | 286 |

| | | | | |
|---|---|---|---|---|
| 128 | 240 | 1844 | 984 | 2828 |
| 123 | 237 | 2458 | 1518 | 3976 |
| 147 | 317 | 4183 | 2531 | 6714 |
| 86 | 113 | 1679 | 1108 | 2787 |
| 16 | 200 | 2113 | 2151 | 4264 |
| 112 | 212 | 3137 | 1729 | 4866 |
| 124 | 271 | 6982 | 3950 | 10932 |
| 15 | 380 | 1772 | 910 | 2682 |
| 26 | 53 | 170 | 90 | 260 |
| 105 | 144 | 1777 | 847 | 2624 |
| 53 | 80 | 563 | 247 | 810 |
| 60 | 115 | 1143 | 664 | 1807 |
| 77 | 154 | 1936 | 1161 | 3097 |
| 39 | 50 | 262 | 166 | 428 |
| 56 | 41 | 408 | 196 | 604 |
| 56 | 41 | 408 | 196 | 604 |
| 178 | 238 | 4197 | 2283 | 6480 |
| 94 | 226 | 2229 | 1130 | 3359 |
| 82 | 76 | 886 | 414 | 1300 |
| 20 | 9 | 49 | 18 | 67 |
| 86 | 114 | 919 | 461 | 1380 |
| 51 | 53 | 414 | 183 | 597 |
| 46 | 51 | 425 | 217 | 642 |
| 38 | 26 | 140 | 64 | 204 |
| 59 | 64 | 423 | 198 | 621 |
| 39 | 27 | 141 | 66 | 207 |
| 26 | 13 | 94 | 43 | 137 |
| 57 | 53 | 355 | 189 | 544 |
| 48 | 40 | 360 | 170 | 530 |
| 69 | 77 | 820 | 371 | 1191 |
| 54 | 49 | 499 | 225 | 724 |
| 96 | 382 | 4731 | 2665 | 7396 |
| 158 | 310 | 4285 | 2142 | 6427 |
| 139 | 457 | 6033 | 2933 | 8966 |
| 59 | 103 | 622 | 306 | 928 |
| 84 | 113 | 1219 | 538 | 1757 |
| 67 | 131 | 666 | 273 | 939 |
| 91 | 100 | 991 | 451 | 1442 |
| 84 | 162 | 1049 | 451 | 1500 |
| 52 | 42 | 268 | 128 | 396 |
| 36 | 31 | 136 | 59 | 195 |
| 60 | 77 | 384 | 184 | 568 |
| 104 | 171 | 1687 | 675 | 2362 |
| 45 | 48 | 232 | 103 | 335 |
| 64 | 106 | 483 | 246 | 729 |
| 48 | 142 | 643 | 319 | 962 |
| 47 | 31 | 237 | 121 | 358 |
| 43 | 44 | 182 | 84 | 266 |
| 57 | 246 | 1143 | 524 | 1667 |
| 82 | 203 | 1486 | 753 | 2239 |
| 79 | 94 | 965 | 437 | 1402 |

| | | | | |
|---|---|---|---|---|
| 129 | 242 | 1630 | 792 | 2422 |
| 175 | 403 | 4839 | 2681 | 7520 |
| 78 | 115 | 839 | 362 | 1201 |
| 48 | 49 | 1155 | 843 | 1998 |
| 39 | 31 | 194 | 107 | 301 |
| 40 | 34 | 221 | 93 | 314 |
| 63 | 248 | 1716 | 1119 | 2835 |
| 74 | 287 | 2935 | 2245 | 5180 |
| 50 | 43 | 317 | 198 | 515 |
| 96 | 120 | 1073 | 456 | 1529 |
| 128 | 191 | 1910 | 729 | 2639 |
| 48 | 104 | 479 | 272 | 751 |
| 70 | 113 | 890 | 376 | 1266 |
| 81 | 83 | 957 | 513 | 1470 |
| 72 | 91 | 1084 | 657 | 1741 |
| 158 | 421 | 3762 | 1926 | 5688 |
| 59 | 70 | 515 | 267 | 782 |
| 104 | 209 | 2097 | 962 | 3059 |
| 106 | 177 | 998 | 546 | 1544 |
| 28 | 22 | 122 | 80 | 202 |
| 69 | 141 | 996 | 488 | 1484 |
| 152 | 289 | 6683 | 3177 | 9860 |
| 139 | 326 | 3530 | 2002 | 5532 |
| 75 | 373 | 1941 | 1289 | 3230 |
| 88 | 153 | 1928 | 907 | 2835 |
| 46 | 39 | 340 | 147 | 487 |
| 80 | 106 | 704 | 375 | 1079 |
| 75 | 126 | 1182 | 606 | 1788 |
| 83 | 93 | 1316 | 801 | 2117 |
| 69 | 85 | 623 | 330 | 953 |
| 33 | 26 | 181 | 64 | 245 |
| 202 | 536 | 7431 | 3646 | 11077 |
| 49 | 68 | 533 | 341 | 874 |
| 77 | 82 | 831 | 460 | 1291 |
| 75 | 120 | 922 | 578 | 1500 |
| 95 | 140 | 1425 | 833 | 2258 |
| 65 | 118 | 976 | 577 | 1553 |
| 46 | 37 | 292 | 126 | 418 |
| 95 | 154 | 1230 | 713 | 1943 |
| 36 | 45 | 430 | 247 | 677 |
| 48 | 73 | 555 | 309 | 864 |
| 60 | 48 | 484 | 219 | 703 |
| 22 | 195 | 687 | 445 | 1132 |
| 96 | 183 | 1948 | 1030 | 2978 |
| 120 | 278 | 1538 | 781 | 2319 |
| 85 | 94 | 1059 | 546 | 1605 |
| 122 | 236 | 2735 | 1680 | 4415 |
| 70 | 104 | 970 | 587 | 1557 |
| 92 | 120 | 916 | 401 | 1317 |
| 43 | 127 | 704 | 394 | 1098 |
| 101 | 141 | 2014 | 1100 | 3114 |

| | | | | |
|---|---|---|---|---|
| 57 | 77 | 564 | 358 | 922 |
| 54 | 25 | 446 | 209 | 655 |
| 45 | 128 | 750 | 425 | 1175 |
| 29 | 26 | 137 | 102 | 239 |
| 43 | 25 | 277 | 93 | 370 |
| 60 | 95 | 602 | 288 | 890 |
| 48 | 46 | 246 | 116 | 362 |
| 157 | 810 | 6236 | 3747 | 9983 |
| 77 | 135 | 1133 | 555 | 1688 |
| 87 | 157 | 1210 | 779 | 1989 |
| 126 | 296 | 3056 | 1431 | 4487 |
| 81 | 122 | 1052 | 697 | 1749 |
| 43 | 41 | 393 | 181 | 574 |
| 111 | 175 | 1436 | 786 | 2222 |
| 169 | 373 | 6241 | 3710 | 9951 |
| 61 | 211 | 754 | 546 | 1300 |
| 79 | 436 | 2374 | 1504 | 3878 |
| 126 | 318 | 2537 | 1500 | 4037 |
| 97 | 357 | 2626 | 1418 | 4044 |
| 87 | 125 | 675 | 358 | 1033 |
| 108 | 184 | 1703 | 952 | 2655 |
| 164 | 305 | 3336 | 1992 | 5328 |
| 84 | 136 | 785 | 499 | 1284 |
| 173 | 280 | 4219 | 2275 | 6494 |
| 91 | 110 | 783 | 377 | 1160 |
| 121 | 262 | 1811 | 1140 | 2951 |
| 31 | 22 | 106 | 67 | 173 |
| 40 | 37 | 292 | 166 | 458 |
| 127 | 271 | 2435 | 1317 | 3752 |
| 75 | 97 | 1140 | 577 | 1717 |
| 104 | 227 | 2186 | 1140 | 3326 |
| 106 | 218 | 1826 | 923 | 2749 |
| 149 | 302 | 3556 | 1588 | 5144 |
| 132 | 269 | 2781 | 1517 | 4298 |
| 66 | 94 | 639 | 324 | 963 |
| 232 | 580 | 8252 | 4194 | 12446 |
| 33 | 34 | 241 | 107 | 348 |
| 47 | 44 | 226 | 134 | 360 |
| 23 | 16 | 79 | 38 | 117 |
| 52 | 49 | 329 | 169 | 498 |
| 33 | 25 | 140 | 80 | 220 |
| 33 | 25 | 140 | 80 | 220 |
| 35 | 20 | 148 | 85 | 233 |
| 12 | 8 | 41 | 31 | 72 |
| 12 | 8 | 41 | 31 | 72 |
| 39 | 63 | 249 | 181 | 430 |
| 37 | 60 | 243 | 180 | 423 |
| 8 | 4 | 19 | 13 | 32 |
| 65 | 82 | 692 | 311 | 1003 |
| 104 | 170 | 1521 | 728 | 2249 |
| 16 | 6 | 31 | 12 | 43 |

| | | | | |
|---|---|---|---|---|
| 37 | 32 | 281 | 126 | 407 |
| 25 | 16 | 147 | 61 | 208 |
| 31 | 29 | 314 | 121 | 435 |
| 26 | 20 | 90 | 41 | 131 |
| 19 | 9 | 45 | 18 | 63 |
| 20 | 7 | 30 | 10 | 40 |
| 26 | 16 | 108 | 46 | 154 |
| 16 | 8 | 61 | 25 | 86 |
| 18 | 10 | 52 | 30 | 82 |
| 36 | 32 | 186 | 86 | 272 |
| 20 | 10 | 52 | 22 | 74 |
| 25 | 12 | 129 | 66 | 195 |
| 42 | 32 | 352 | 136 | 488 |
| 26 | 14 | 71 | 42 | 113 |
| 28 | 19 | 109 | 57 | 166 |
| 23 | 10 | 98 | 41 | 139 |
| 18 | 454 | 710 | 547 | 1257 |
| 92 | 140 | 965 | 495 | 1460 |
| 71 | 152 | 1372 | 860 | 2232 |
| 95 | 170 | 1912 | 999 | 2911 |
| 78 | 136 | 1242 | 636 | 1878 |
| 68 | 90 | 641 | 339 | 980 |
| 66 | 60 | 812 | 403 | 1215 |
| 47 | 72 | 637 | 317 | 954 |
| 16 | 8 | 29 | 14 | 43 |
| 33 | 26 | 156 | 73 | 229 |
| 150 | 502 | 4257 | 2231 | 6488 |
| 119 | 151 | 2254 | 1254 | 3508 |
| 87 | 112 | 805 | 370 | 1175 |
| 49 | 55 | 440 | 235 | 675 |
| 80 | 157 | 1189 | 531 | 1720 |
| 82 | 146 | 949 | 434 | 1383 |
| 35 | 35 | 278 | 156 | 434 |
| 36 | 25 | 154 | 60 | 214 |
| 197 | 594 | 7326 | 3819 | 11145 |
| 48 | 51 | 331 | 137 | 468 |
| 75 | 185 | 1133 | 526 | 1659 |
| 66 | 67 | 814 | 418 | 1232 |
| 44 | 31 | 187 | 91 | 278 |
| 64 | 78 | 485 | 244 | 729 |
| 70 | 158 | 991 | 485 | 1476 |
| 67 | 91 | 542 | 258 | 800 |
| 39 | 100 | 456 | 254 | 710 |
| 24 | 179 | 410 | 238 | 648 |
| 81 | 113 | 1046 | 385 | 1431 |
| 20 | 366 | 663 | 514 | 1177 |
| 88 | 226 | 3518 | 2048 | 5566 |
| 27 | 11 | 96 | 48 | 144 |
| 17 | 9 | 33 | 17 | 50 |
| 90 | 116 | 1336 | 639 | 1975 |
| 87 | 128 | 1022 | 494 | 1516 |

| | | | | |
|---|---|---|---|---|
| 158 | 316 | 2970 | 1729 | 4699 |
| 110 | 236 | 2208 | 1224 | 3432 |
| 38 | 50 | 154 | 66 | 220 |
| 16 | 63 | 460 | 305 | 765 |
| 10 | 20 | 47 | 29 | 76 |
| 135 | 275 | 2543 | 1227 | 3770 |
| 18 | 7 | 38 | 11 | 49 |
| 56 | 115 | 741 | 359 | 1100 |
| 44 | 35 | 307 | 157 | 464 |
| 50 | 79 | 456 | 242 | 698 |
| 46 | 42 | 286 | 126 | 412 |
| 67 | 60 | 657 | 292 | 949 |
| 18 | 11 | 36 | 18 | 54 |
| 53 | 61 | 516 | 284 | 800 |
| 19 | 17 | 95 | 55 | 150 |
| 70 | 75 | 677 | 309 | 986 |
| 27 | 26 | 116 | 56 | 172 |
| 51 | 49 | 409 | 183 | 592 |
| 25 | 10 | 84 | 30 | 114 |
| 22 | 10 | 64 | 19 | 83 |
| 29 | 16 | 97 | 41 | 138 |
| 28 | 16 | 79 | 32 | 111 |
| 20 | 9 | 57 | 19 | 76 |
| 33 | 15 | 106 | 40 | 146 |
| 22 | 13 | 80 | 30 | 110 |
| 24 | 15 | 47 | 21 | 68 |
| 24 | 16 | 48 | 22 | 70 |
| 24 | 15 | 47 | 21 | 68 |
| 26 | 14 | 77 | 34 | 111 |
| 46 | 26 | 241 | 97 | 338 |
| 115 | 154 | 1487 | 688 | 2175 |
| 132 | 225 | 2769 | 1228 | 3997 |
| 76 | 89 | 1003 | 463 | 1466 |
| 161 | 269 | 2913 | 1352 | 4265 |
| 25 | 17 | 67 | 25 | 92 |
| 108 | 138 | 1631 | 697 | 2328 |
| 36 | 26 | 215 | 131 | 346 |
| 52 | 37 | 318 | 149 | 467 |
| 49 | 30 | 254 | 118 | 372 |
| 49 | 30 | 254 | 118 | 372 |
| 50 | 51 | 369 | 166 | 535 |
| 32 | 20 | 114 | 51 | 165 |
| 31 | 19 | 89 | 34 | 123 |
| 21 | 7 | 38 | 15 | 53 |
| 30 | 19 | 98 | 34 | 132 |
| 129 | 275 | 2656 | 1348 | 4004 |
| 128 | 170 | 2143 | 927 | 3070 |
| 14 | 71 | 180 | 83 | 263 |
| 125 | 168 | 1487 | 677 | 2164 |
| 156 | 304 | 3578 | 1655 | 5233 |
| 123 | 129 | 1800 | 823 | 2623 |

| | | | | |
|---|---|---|---|---|
| 117 | 129 | 1269 | 643 | 1912 |
| 99 | 87 | 1375 | 681 | 2056 |
| 147 | 182 | 2382 | 1146 | 3528 |
| 65 | 50 | 386 | 182 | 568 |
| 63 | 133 | 619 | 380 | 999 |
| 152 | 189 | 2418 | 1020 | 3438 |
| 15 | 22 | 77 | 52 | 129 |
| 72 | 58 | 420 | 188 | 608 |
| 10 | 68 | 144 | 81 | 225 |
| 134 | 149 | 1811 | 838 | 2649 |
| 43 | 25 | 421 | 186 | 607 |
| 62 | 79 | 761 | 388 | 1149 |
| 46 | 51 | 230 | 123 | 353 |
| 127 | 172 | 1671 | 759 | 2430 |
| 79 | 142 | 1614 | 865 | 2479 |
| 44 | 31 | 188 | 80 | 268 |
| 21 | 12 | 116 | 39 | 155 |
| 80 | 94 | 683 | 279 | 962 |
| 121 | 147 | 2047 | 1128 | 3175 |
| 106 | 149 | 2303 | 1049 | 3352 |
| 49 | 37 | 282 | 107 | 389 |
| 77 | 69 | 871 | 393 | 1264 |
| 35 | 29 | 177 | 68 | 245 |
| 125 | 154 | 1586 | 765 | 2351 |
| 54 | 34 | 483 | 293 | 776 |
| 44 | 26 | 182 | 104 | 286 |
| 36 | 42 | 210 | 84 | 294 |
| 94 | 104 | 1461 | 681 | 2142 |
| 35 | 20 | 105 | 38 | 143 |
| 53 | 52 | 540 | 265 | 805 |
| 105 | 109 | 1321 | 529 | 1850 |
| 89 | 157 | 1488 | 842 | 2330 |
| 63 | 59 | 435 | 200 | 635 |
| 34 | 36 | 208 | 148 | 356 |
| 106 | 228 | 2295 | 1179 | 3474 |
| 98 | 194 | 1389 | 681 | 2070 |
| 127 | 241 | 2813 | 1614 | 4427 |
| 14 | 4 | 22 | 8 | 30 |
| 36 | 25 | 135 | 64 | 199 |
| 7 | 87 | 192 | 159 | 351 |
| 15 | 9 | 41 | 17 | 58 |
| 25 | 18 | 69 | 33 | 102 |
| 43 | 35 | 194 | 80 | 274 |
| 43 | 31 | 233 | 98 | 331 |
| 122 | 289 | 3135 | 1945 | 5080 |
| 47 | 40 | 262 | 116 | 378 |
| 59 | 59 | 472 | 238 | 710 |
| 79 | 91 | 843 | 389 | 1232 |
| 51 | 55 | 418 | 185 | 603 |
| 91 | 136 | 1237 | 646 | 1883 |
| 72 | 78 | 543 | 218 | 761 |

| | | | | |
|---|---|---|---|---|
| 120 | 163 | 1587 | 747 | 2334 |
| 75 | 185 | 2220 | 1145 | 3365 |
| 67 | 67 | 495 | 212 | 707 |
| 98 | 106 | 920 | 408 | 1328 |
| 35 | 17 | 117 | 50 | 167 |
| 74 | 113 | 1341 | 705 | 2046 |
| 36 | 26 | 187 | 71 | 258 |
| 74 | 85 | 824 | 385 | 1209 |
| 90 | 133 | 1231 | 738 | 1969 |
| 80 | 111 | 1009 | 493 | 1502 |
| 52 | 58 | 560 | 302 | 862 |
| 159 | 314 | 3988 | 1829 | 5817 |
| 122 | 186 | 2084 | 1019 | 3103 |
| 173 | 266 | 2619 | 1227 | 3846 |
| 47 | 40 | 254 | 108 | 362 |
| 54 | 64 | 351 | 159 | 510 |
| 72 | 95 | 693 | 371 | 1064 |
| 31 | 25 | 177 | 84 | 261 |
| 28 | 20 | 113 | 56 | 169 |
| 157 | 372 | 4227 | 2054 | 6281 |
| 86 | 145 | 1589 | 769 | 2358 |
| 50 | 71 | 521 | 278 | 799 |
| 48 | 42 | 299 | 144 | 443 |
| 37 | 36 | 243 | 106 | 349 |
| 78 | 94 | 1165 | 507 | 1672 |
| 54 | 69 | 645 | 272 | 917 |
| 58 | 81 | 870 | 394 | 1264 |
| 65 | 117 | 858 | 397 | 1255 |
| 33 | 23 | 224 | 105 | 329 |
| 56 | 75 | 650 | 277 | 927 |
| 73 | 61 | 509 | 203 | 712 |
| 50 | 50 | 386 | 156 | 542 |
| 88 | 219 | 2080 | 1247 | 3327 |
| 112 | 282 | 3000 | 1688 | 4688 |
| 98 | 171 | 1976 | 1158 | 3134 |
| 76 | 107 | 1612 | 990 | 2602 |
| 121 | 298 | 4278 | 2495 | 6773 |
| 85 | 131 | 836 | 377 | 1213 |
| 131 | 257 | 2737 | 1244 | 3981 |
| 88 | 134 | 1040 | 576 | 1616 |
| 162 | 278 | 2631 | 1288 | 3919 |
| 154 | 298 | 3380 | 1655 | 5035 |
| 23 | 13 | 47 | 19 | 66 |
| 134 | 290 | 3328 | 1430 | 4758 |
| 146 | 218 | 2078 | 959 | 3037 |
| 100 | 93 | 728 | 320 | 1048 |
| 90 | 185 | 1808 | 929 | 2737 |
| 94 | 153 | 1736 | 784 | 2520 |
| 48 | 35 | 244 | 107 | 351 |
| 58 | 48 | 402 | 173 | 575 |
| 35 | 32 | 224 | 113 | 337 |

| | | | | |
|---|---|---|---|---|
| 68 | 121 | 1100 | 513 | 1613 |
| 15 | 5 | 20 | 7 | 27 |
| 74 | 160 | 1780 | 953 | 2733 |
| 31 | 35 | 168 | 87 | 255 |
| 30 | 21 | 256 | 102 | 358 |
| 17 | 6 | 29 | 11 | 40 |
| 22 | 8 | 44 | 18 | 62 |
| 24 | 19 | 98 | 57 | 155 |
| 18 | 7 | 51 | 20 | 71 |
| 47 | 63 | 418 | 197 | 615 |
| 24 | 13 | 66 | 32 | 98 |
| 30 | 27 | 127 | 52 | 179 |
| 22 | 12 | 49 | 18 | 67 |
| 16 | 7 | 25 | 8 | 33 |
| 17 | 6 | 33 | 12 | 45 |
| 21 | 10 | 76 | 40 | 116 |
| 61 | 41 | 422 | 176 | 598 |
| 12 | 3 | 20 | 3 | 23 |
| 22 | 9 | 45 | 19 | 64 |
| 22 | 9 | 45 | 19 | 64 |
| 50 | 104 | 788 | 383 | 1171 |
| 54 | 99 | 1216 | 583 | 1799 |
| 26 | 20 | 93 | 48 | 141 |
| 19 | 9 | 55 | 23 | 78 |
| 14 | 5 | 27 | 12 | 39 |
| 21 | 10 | 49 | 20 | 69 |
| 31 | 32 | 147 | 64 | 211 |
| 24 | 10 | 74 | 34 | 108 |
| 15 | 3 | 24 | 6 | 30 |
| 22 | 9 | 57 | 26 | 83 |
| 19 | 9 | 65 | 28 | 93 |
| 25 | 15 | 109 | 44 | 153 |
| 10 | 3 | 12 | 4 | 16 |
| 27 | 14 | 77 | 37 | 114 |
| 19 | 6 | 43 | 16 | 59 |
| 53 | 35 | 427 | 206 | 633 |
| 29 | 12 | 78 | 27 | 105 |
| 24 | 15 | 64 | 29 | 93 |
| 29 | 21 | 116 | 44 | 160 |
| 24 | 15 | 83 | 35 | 118 |
| 17 | 6 | 32 | 11 | 43 |
| 14 | 5 | 23 | 7 | 30 |
| 21 | 9 | 40 | 18 | 58 |
| 18 | 6 | 34 | 13 | 47 |
| 16 | 6 | 34 | 13 | 47 |
| 27 | 13 | 75 | 31 | 106 |
| 39 | 18 | 149 | 50 | 199 |
| 20 | 12 | 44 | 25 | 69 |
| 51 | 44 | 336 | 174 | 510 |
| 26 | 22 | 108 | 54 | 162 |
| 36 | 22 | 147 | 65 | 212 |

| | | | | |
|---|---|---|---|---|
| 27 | 18 | 152 | 59 | 211 |
| 65 | 82 | 536 | 302 | 838 |
| 43 | 31 | 187 | 88 | 275 |
| 76 | 91 | 711 | 309 | 1020 |
| 18 | 17 | 67 | 38 | 105 |
| 41 | 27 | 288 | 161 | 449 |
| 19 | 12 | 42 | 26 | 68 |
| 27 | 12 | 63 | 26 | 89 |
| 18 | 9 | 30 | 13 | 43 |
| 34 | 24 | 141 | 80 | 221 |
| 22 | 10 | 49 | 16 | 65 |
| 20 | 14 | 52 | 25 | 77 |
| 57 | 42 | 392 | 174 | 566 |
| 42 | 31 | 236 | 114 | 350 |
| 50 | 42 | 291 | 142 | 433 |
| 58 | 70 | 572 | 322 | 894 |
| 36 | 45 | 231 | 146 | 377 |
| 24 | 11 | 80 | 31 | 111 |
| 42 | 27 | 149 | 62 | 211 |
| 22 | 12 | 65 | 27 | 92 |
| 26 | 16 | 66 | 41 | 107 |
| 27 | 15 | 79 | 32 | 111 |
| 35 | 16 | 185 | 71 | 256 |
| 29 | 12 | 66 | 25 | 91 |
| 30 | 22 | 110 | 45 | 155 |
| 85 | 134 | 1346 | 748 | 2094 |
| 32 | 19 | 232 | 147 | 379 |
| 45 | 67 | 271 | 148 | 419 |
| 51 | 68 | 678 | 308 | 986 |
| 84 | 117 | 981 | 506 | 1487 |
| 126 | 225 | 2242 | 1031 | 3273 |
| 70 | 52 | 650 | 281 | 931 |
| 98 | 170 | 1510 | 797 | 2307 |
| 77 | 99 | 549 | 252 | 801 |
| 7 | 47 | 81 | 57 | 138 |
| 88 | 94 | 951 | 423 | 1374 |
| 15 | 119 | 425 | 234 | 659 |
| 49 | 68 | 325 | 160 | 485 |
| 137 | 313 | 3418 | 1723 | 5141 |
| 56 | 100 | 880 | 434 | 1314 |
| 112 | 331 | 3234 | 1833 | 5067 |
| 70 | 110 | 785 | 518 | 1303 |
| 120 | 162 | 1667 | 743 | 2410 |
| 156 | 379 | 5117 | 2824 | 7941 |
| 27 | 141 | 239 | 188 | 427 |
| 9 | 6 | 23 | 6 | 29 |
| 114 | 221 | 2010 | 1039 | 3049 |
| 59 | 93 | 745 | 383 | 1128 |
| 7 | 7 | 12 | 7 | 19 |
| 11 | 13 | 32 | 24 | 56 |
| 93 | 161 | 1684 | 728 | 2412 |

| | | | | |
|---|---|---|---|---|
| 72 | 215 | 2575 | 1271 | 3846 |
| 70 | 164 | 1707 | 843 | 2550 |
| 35 | 35 | 171 | 93 | 264 |
| 157 | 297 | 3898 | 1672 | 5570 |
| 80 | 145 | 1049 | 645 | 1694 |
| 51 | 55 | 322 | 171 | 493 |
| 59 | 68 | 338 | 186 | 524 |
| 42 | 43 | 222 | 88 | 310 |
| 64 | 78 | 852 | 497 | 1349 |
| 38 | 32 | 136 | 69 | 205 |
| 10 | 10 | 20 | 10 | 30 |
| 86 | 216 | 1509 | 926 | 2435 |
| 90 | 180 | 1368 | 631 | 1999 |
| 80 | 138 | 953 | 418 | 1371 |
| 87 | 130 | 999 | 440 | 1439 |
| 25 | 11 | 69 | 26 | 95 |
| 62 | 77 | 558 | 294 | 852 |
| 8 | 5 | 9 | 5 | 14 |
| 8 | 4 | 9 | 4 | 13 |
| 55 | 58 | 367 | 214 | 581 |
| 71 | 119 | 556 | 317 | 873 |
| 9 | 16 | 41 | 26 | 67 |
| 59 | 196 | 1977 | 1282 | 3259 |
| 71 | 117 | 552 | 315 | 867 |
| 9 | 16 | 41 | 26 | 67 |
| 59 | 198 | 2003 | 1298 | 3301 |
| 140 | 372 | 3825 | 1825 | 5650 |
| 31 | 19 | 154 | 63 | 217 |
| 28 | 23 | 125 | 48 | 173 |
| 45 | 48 | 297 | 152 | 449 |
| 61 | 95 | 875 | 340 | 1215 |
| 98 | 183 | 2373 | 1490 | 3863 |
| 69 | 256 | 1568 | 811 | 2379 |
| 44 | 108 | 653 | 436 | 1089 |
| 36 | 46 | 184 | 112 | 296 |
| 76 | 124 | 1349 | 812 | 2161 |
| 26 | 108 | 242 | 164 | 406 |
| 11 | 25 | 50 | 47 | 97 |
| 100 | 225 | 1672 | 784 | 2456 |
| 89 | 138 | 956 | 367 | 1323 |
| 112 | 298 | 3261 | 1860 | 5121 |
| 103 | 305 | 3152 | 2134 | 5286 |
| 35 | 59 | 1400 | 790 | 2190 |
| 42 | 82 | 334 | 251 | 585 |
| 134 | 400 | 3663 | 2315 | 5978 |
| 61 | 101 | 937 | 637 | 1574 |
| 11 | 30 | 63 | 30 | 93 |
| 69 | 82 | 543 | 293 | 836 |
| 114 | 259 | 2647 | 1591 | 4238 |
| 101 | 185 | 1747 | 1013 | 2760 |
| 84 | 108 | 720 | 348 | 1068 |

| | | | | |
|---|---|---|---|---|
| 6 | 4 | 11 | 4 | 15 |
| 91 | 177 | 1047 | 626 | 1673 |
| 25 | 19 | 84 | 47 | 131 |
| 22 | 15 | 82 | 42 | 124 |
| 21 | 9 | 51 | 14 | 65 |
| 51 | 40 | 318 | 136 | 454 |
| 32 | 21 | 187 | 73 | 260 |
| 46 | 30 | 320 | 127 | 447 |
| 23 | 17 | 65 | 25 | 90 |
| 47 | 37 | 241 | 97 | 338 |
| 46 | 50 | 451 | 250 | 701 |
| 26 | 18 | 64 | 27 | 91 |
| 24 | 18 | 64 | 27 | 91 |
| 24 | 18 | 64 | 27 | 91 |
| 35 | 27 | 128 | 57 | 185 |
| 24 | 17 | 64 | 27 | 91 |
| 25 | 14 | 57 | 19 | 76 |
| 55 | 43 | 460 | 180 | 640 |
| 26 | 18 | 64 | 27 | 91 |
| 32 | 22 | 90 | 37 | 127 |
| 26 | 17 | 64 | 27 | 91 |
| 28 | 14 | 66 | 24 | 90 |
| 30 | 20 | 87 | 35 | 122 |
| 24 | 14 | 58 | 21 | 79 |
| 25 | 8 | 47 | 16 | 63 |
| 23 | 8 | 41 | 15 | 56 |
| 20 | 15 | 46 | 25 | 71 |
| 48 | 68 | 358 | 193 | 551 |
| 37 | 38 | 341 | 104 | 445 |
| 40 | 21 | 175 | 52 | 227 |
| 59 | 69 | 530 | 221 | 751 |
| 59 | 114 | 1261 | 674 | 1935 |
| 48 | 65 | 394 | 173 | 567 |
| 36 | 25 | 98 | 40 | 138 |
| 55 | 54 | 379 | 140 | 519 |
| 46 | 50 | 227 | 106 | 333 |
| 99 | 174 | 2323 | 1146 | 3469 |
| 34 | 38 | 362 | 147 | 509 |
| 29 | 29 | 138 | 68 | 206 |
| 35 | 31 | 262 | 88 | 350 |
| 44 | 43 | 492 | 168 | 660 |
| 66 | 69 | 562 | 204 | 766 |
| 71 | 81 | 723 | 301 | 1024 |
| 38 | 34 | 215 | 90 | 305 |
| 109 | 279 | 3661 | 1900 | 5561 |
| 55 | 72 | 643 | 307 | 950 |
| 57 | 38 | 500 | 242 | 742 |
| 69 | 183 | 1457 | 670 | 2127 |
| 8 | 34 | 266 | 259 | 525 |
| 11 | 55 | 109 | 61 | 170 |
| 40 | 29 | 197 | 68 | 265 |

| | | | | |
|---|---|---|---|---|
| 44 | 30 | 155 | 63 | 218 |
| 64 | 73 | 734 | 341 | 1075 |
| 45 | 46 | 347 | 181 | 528 |
| 51 | 111 | 885 | 380 | 1265 |
| 51 | 32 | 309 | 154 | 463 |
| 81 | 79 | 937 | 457 | 1394 |
| 86 | 81 | 677 | 362 | 1039 |
| 86 | 90 | 789 | 335 | 1124 |
| 102 | 117 | 759 | 346 | 1105 |
| 15 | 243 | 569 | 253 | 822 |
| 109 | 120 | 1362 | 614 | 1976 |
| 62 | 51 | 468 | 216 | 684 |
| 33 | 16 | 112 | 68 | 180 |
| 57 | 51 | 596 | 279 | 875 |
| 129 | 174 | 1462 | 679 | 2141 |
| 13 | 4 | 25 | 8 | 33 |
| 37 | 20 | 172 | 61 | 233 |
| 25 | 13 | 139 | 50 | 189 |
| 56 | 75 | 822 | 279 | 1101 |
| 64 | 61 | 553 | 260 | 813 |
| 167 | 235 | 2047 | 927 | 2974 |
| 22 | 13 | 95 | 56 | 151 |
| 9 | 18 | 31 | 19 | 50 |
| 58 | 32 | 384 | 149 | 533 |
| 80 | 91 | 686 | 301 | 987 |
| 55 | 49 | 323 | 130 | 453 |
| 43 | 26 | 198 | 96 | 294 |
| 34 | 22 | 130 | 67 | 197 |
| 28 | 15 | 81 | 26 | 107 |
| 39 | 27 | 142 | 53 | 195 |
| 26 | 15 | 66 | 32 | 98 |
| 85 | 104 | 1525 | 759 | 2284 |
| 45 | 90 | 1539 | 896 | 2435 |
| 36 | 43 | 721 | 482 | 1203 |
| 48 | 70 | 829 | 491 | 1320 |
| 130 | 501 | 5172 | 2940 | 8112 |
| 35 | 82 | 257 | 177 | 434 |
| 50 | 23 | 234 | 110 | 344 |
| 67 | 41 | 443 | 215 | 658 |
| 19 | 9 | 33 | 11 | 44 |
| 57 | 42 | 377 | 229 | 606 |
| 65 | 38 | 692 | 364 | 1056 |
| 64 | 37 | 398 | 171 | 569 |
| 93 | 247 | 2213 | 1008 | 3221 |
| 67 | 88 | 984 | 599 | 1583 |
| 45 | 52 | 296 | 145 | 441 |
| 72 | 438 | 3021 | 1697 | 4718 |
| 80 | 96 | 617 | 278 | 895 |
| 59 | 43 | 282 | 148 | 430 |
| 101 | 323 | 2910 | 1539 | 4449 |
| 88 | 257 | 1813 | 951 | 2764 |

| | | | | |
|---|---|---|---|---|
| 60 | 57 | 554 | 299 | 853 |
| 146 | 259 | 3156 | 1494 | 4650 |
| 55 | 51 | 244 | 130 | 374 |
| 51 | 53 | 289 | 140 | 429 |
| 69 | 80 | 586 | 281 | 867 |
| 37 | 78 | 1682 | 1621 | 3303 |
| 66 | 60 | 509 | 233 | 742 |
| 96 | 143 | 1467 | 743 | 2210 |
| 41 | 368 | 2553 | 951 | 3504 |
| 53 | 55 | 541 | 325 | 866 |
| 138 | 238 | 4475 | 2280 | 6755 |
| 132 | 257 | 2143 | 993 | 3136 |
| 104 | 107 | 1221 | 638 | 1859 |
| 52 | 39 | 589 | 313 | 902 |
| 33 | 23 | 114 | 47 | 161 |
| 58 | 48 | 328 | 148 | 476 |
| 35 | 34 | 224 | 90 | 314 |
| 59 | 39 | 421 | 172 | 593 |
| 78 | 134 | 1373 | 771 | 2144 |
| 14 | 8 | 28 | 13 | 41 |
| 82 | 111 | 976 | 518 | 1494 |
| 78 | 78 | 641 | 314 | 955 |
| 19 | 11 | 66 | 29 | 95 |
| 35 | 30 | 127 | 56 | 183 |
| 66 | 80 | 829 | 382 | 1211 |
| 132 | 250 | 2059 | 890 | 2949 |
| 159 | 358 | 4066 | 1897 | 5963 |
| 133 | 301 | 3154 | 1522 | 4676 |
| 42 | 29 | 194 | 80 | 274 |
| 55 | 47 | 403 | 172 | 575 |
| 36 | 24 | 212 | 81 | 293 |
| 49 | 32 | 180 | 82 | 262 |
| 36 | 40 | 298 | 118 | 416 |
| 30 | 31 | 362 | 156 | 518 |
| 23 | 14 | 107 | 46 | 153 |
| 72 | 56 | 587 | 287 | 874 |
| 31 | 18 | 81 | 37 | 118 |
| 58 | 44 | 378 | 184 | 562 |
| 56 | 44 | 280 | 127 | 407 |
| 45 | 39 | 215 | 96 | 311 |
| 22 | 13 | 72 | 27 | 99 |
| 37 | 26 | 174 | 80 | 254 |
| 57 | 96 | 643 | 380 | 1023 |
| 46 | 73 | 424 | 237 | 661 |
| 38 | 28 | 148 | 63 | 211 |
| 86 | 149 | 1606 | 1003 | 2609 |
| 86 | 135 | 1067 | 590 | 1657 |
| 12 | 35 | 56 | 43 | 99 |
| 11 | 3 | 24 | 10 | 34 |
| 19 | 7 | 39 | 16 | 55 |
| 65 | 73 | 463 | 217 | 680 |

| | | | | |
|---|---|---|---|---|
| 56 | 53 | 399 | 166 | 565 |
| 11 | 4 | 14 | 8 | 22 |
| 52 | 48 | 316 | 148 | 464 |
| 50 | 42 | 206 | 89 | 295 |
| 65 | 57 | 495 | 224 | 719 |
| 45 | 40 | 162 | 77 | 239 |
| 58 | 41 | 236 | 106 | 342 |
| 55 | 37 | 208 | 96 | 304 |
| 105 | 169 | 1286 | 627 | 1913 |
| 118 | 232 | 2364 | 1100 | 3464 |
| 13 | 35 | 71 | 44 | 115 |
| 26 | 12 | 56 | 28 | 84 |
| 63 | 84 | 552 | 254 | 806 |
| 43 | 41 | 285 | 132 | 417 |
| 125 | 258 | 2286 | 1121 | 3407 |
| 93 | 172 | 1546 | 748 | 2294 |
| 116 | 216 | 2821 | 1357 | 4178 |
| 129 | 234 | 2121 | 991 | 3112 |
| 32 | 24 | 145 | 64 | 209 |
| 66 | 46 | 465 | 210 | 675 |
| 34 | 31 | 248 | 94 | 342 |
| 114 | 161 | 1879 | 866 | 2745 |
| 50 | 29 | 247 | 111 | 358 |
| 89 | 68 | 749 | 325 | 1074 |
| 105 | 112 | 1323 | 532 | 1855 |
| 97 | 125 | 726 | 349 | 1075 |
| 114 | 146 | 2408 | 1349 | 3757 |
| 122 | 196 | 1506 | 721 | 2227 |
| 158 | 244 | 2210 | 1038 | 3248 |
| 116 | 169 | 1690 | 782 | 2472 |
| 23 | 10 | 65 | 22 | 87 |
| 87 | 156 | 858 | 431 | 1289 |
| 72 | 55 | 562 | 229 | 791 |
| 175 | 205 | 2203 | 651 | 2854 |
| 135 | 126 | 1021 | 368 | 1389 |
| 181 | 266 | 3195 | 1576 | 4771 |
| 13 | 122 | 499 | 343 | 842 |
| 8 | 9 | 19 | 9 | 28 |
| 84 | 69 | 697 | 305 | 1002 |
| 152 | 240 | 2261 | 989 | 3250 |
| 130 | 180 | 2070 | 1016 | 3086 |
| 124 | 200 | 2436 | 1072 | 3508 |
| 80 | 68 | 621 | 237 | 858 |
| 187 | 225 | 2196 | 907 | 3103 |
| 150 | 202 | 2315 | 1214 | 3529 |
| 72 | 199 | 872 | 492 | 1364 |
| 101 | 108 | 876 | 376 | 1252 |
| 144 | 164 | 1901 | 916 | 2817 |
| 130 | 162 | 1766 | 884 | 2650 |
| 75 | 78 | 1313 | 599 | 1912 |
| 125 | 161 | 2628 | 1273 | 3901 |

| | | | | |
|---|---|---|---|---|
| 96 | 85 | 1009 | 516 | 1525 |
| 93 | 102 | 770 | 378 | 1148 |
| 124 | 185 | 1656 | 813 | 2469 |
| 31 | 16 | 85 | 37 | 122 |
| 55 | 84 | 756 | 372 | 1128 |
| 101 | 178 | 2070 | 1251 | 3321 |
| 123 | 302 | 3149 | 1631 | 4780 |
| 44 | 37 | 274 | 103 | 377 |
| 33 | 35 | 248 | 139 | 387 |
| 34 | 32 | 179 | 84 | 263 |
| 38 | 38 | 249 | 108 | 357 |
| 79 | 96 | 1169 | 507 | 1676 |
| 55 | 55 | 522 | 235 | 757 |
| 57 | 80 | 829 | 376 | 1205 |
| 66 | 119 | 864 | 399 | 1263 |
| 34 | 25 | 230 | 107 | 337 |
| 48 | 42 | 247 | 108 | 355 |
| 42 | 56 | 480 | 248 | 728 |
| 70 | 69 | 737 | 370 | 1107 |
| 67 | 112 | 1438 | 859 | 2297 |
| 41 | 31 | 188 | 88 | 276 |
| 73 | 146 | 1377 | 642 | 2019 |
| 62 | 81 | 690 | 326 | 1016 |
| 31 | 30 | 137 | 56 | 193 |
| 53 | 66 | 410 | 194 | 604 |
| 69 | 118 | 738 | 315 | 1053 |
| 92 | 103 | 820 | 364 | 1184 |
| 50 | 40 | 306 | 146 | 452 |
| 121 | 196 | 2037 | 971 | 3008 |
| 89 | 204 | 1526 | 715 | 2241 |
| 81 | 92 | 854 | 363 | 1217 |
| 73 | 131 | 1324 | 624 | 1948 |
| 94 | 185 | 2149 | 1209 | 3358 |
| 94 | 278 | 3056 | 1497 | 4553 |
| 76 | 121 | 1216 | 712 | 1928 |
| 60 | 92 | 1007 | 592 | 1599 |

# Appendix B

**Token Counts Of CMPSC 541 Programs**

| $\eta_1$ | $\eta_2$ | $N_1$ | $N_2$ | $N$ |
|---|---|---|---|---|
| 32 | 22 | 117 | 72 | 189 |
| 32 | 22 | 127 | 79 | 206 |
| 32 | 22 | 210 | 136 | 346 |
| 29 | 21 | 418 | 195 | 613 |
| 97 | 219 | 4568 | 2794 | 7362 |
| 31 | 27 | 186 | 82 | 268 |
| 46 | 48 | 828 | 583 | 1411 |
| 28 | 32 | 265 | 189 | 454 |
| 56 | 131 | 1332 | 788 | 2120 |
| 43 | 56 | 2764 | 2023 | 4787 |
| 41 | 75 | 620 | 367 | 987 |
| 31 | 21 | 146 | 62 | 208 |
| 33 | 36 | 235 | 131 | 366 |
| 24 | 32 | 171 | 124 | 295 |
| 29 | 22 | 76 | 41 | 117 |
| 32 | 37 | 177 | 89 | 266 |
| 22 | 25 | 99 | 74 | 173 |
| 16 | 28 | 477 | 314 | 791 |
| 22 | 27 | 163 | 124 | 287 |
| 26 | 19 | 110 | 54 | 164 |
| 70 | 433 | 3684 | 3039 | 6723 |
| 35 | 56 | 353 | 187 | 540 |
| 25 | 20 | 82 | 31 | 113 |
| 18 | 13 | 242 | 157 | 399 |
| 22 | 31 | 195 | 132 | 327 |
| 19 | 11 | 85 | 32 | 117 |
| 22 | 25 | 99 | 74 | 173 |
| 29 | 60 | 309 | 186 | 495 |
| 55 | 96 | 1304 | 584 | 1888 |
| 40 | 34 | 272 | 153 | 425 |
| 40 | 39 | 315 | 179 | 494 |
| 20 | 47 | 485 | 330 | 815 |
| 26 | 22 | 237 | 163 | 400 |
| 76 | 99 | 1555 | 853 | 2408 |
| 29 | 13 | 104 | 54 | 158 |
| 37 | 32 | 207 | 88 | 295 |
| 32 | 32 | 384 | 226 | 610 |
| 22 | 39 | 400 | 300 | 700 |
| 17 | 11 | 160 | 110 | 270 |
| 31 | 28 | 342 | 165 | 507 |
| 21 | 17 | 58 | 39 | 97 |
| 42 | 56 | 334 | 162 | 496 |
| 23 | 20 | 124 | 87 | 211 |
| 44 | 41 | 311 | 173 | 484 |
| 14 | 5 | 23 | 11 | 34 |
| 22 | 25 | 149 | 114 | 263 |
| 43 | 52 | 494 | 297 | 791 |
| 34 | 47 | 369 | 260 | 629 |
| 28 | 27 | 123 | 89 | 212 |

| | | | | |
|---|---|---|---|---|
| 13 | 5 | 26 | 10 | 36 |
| 28 | 25 | 107 | 56 | 163 |
| 15 | 28 | 121 | 81 | 202 |
| 24 | 14 | 89 | 47 | 136 |
| 42 | 39 | 215 | 137 | 352 |
| 39 | 31 | 258 | 121 | 379 |
| 32 | 26 | 187 | 90 | 277 |
| 14 | 9 | 26 | 15 | 41 |
| 28 | 21 | 112 | 62 | 174 |
| 38 | 38 | 195 | 118 | 313 |
| 25 | 15 | 168 | 62 | 230 |
| 44 | 79 | 577 | 293 | 870 |
| 30 | 30 | 191 | 76 | 267 |
| 21 | 18 | 109 | 47 | 156 |
| 46 | 114 | 3594 | 1138 | 4732 |
| 28 | 33 | 137 | 81 | 218 |
| 28 | 43 | 190 | 132 | 322 |
| 28 | 23 | 156 | 61 | 217 |
| 14 | 8 | 28 | 11 | 39 |
| 15 | 18 | 116 | 47 | 163 |
| 21 | 20 | 88 | 54 | 142 |
| 18 | 17 | 124 | 67 | 191 |
| 28 | 28 | 216 | 116 | 332 |
| 39 | 36 | 303 | 148 | 451 |
| 28 | 19 | 99 | 46 | 145 |
| 29 | 23 | 113 | 66 | 179 |
| 21 | 41 | 244 | 177 | 421 |
| 44 | 68 | 609 | 294 | 903 |
| 18 | 11 | 63 | 25 | 88 |
| 17 | 13 | 53 | 33 | 86 |
| 79 | 137 | 1665 | 802 | 2467 |
| 37 | 49 | 289 | 155 | 444 |
| 21 | 22 | 83 | 53 | 136 |
| 47 | 74 | 790 | 422 | 1212 |
| 31 | 48 | 190 | 130 | 320 |
| 18 | 37 | 199 | 147 | 346 |
| 47 | 70 | 658 | 351 | 1009 |
| 54 | 88 | 592 | 293 | 885 |
| 59 | 139 | 2698 | 1382 | 4080 |
| 65 | 128 | 1900 | 1035 | 2935 |
| 26 | 37 | 337 | 131 | 468 |
| 48 | 73 | 743 | 327 | 1070 |
| 45 | 58 | 521 | 299 | 820 |
| 45 | 62 | 543 | 314 | 857 |
| 14 | 8 | 82 | 56 | 138 |
| 38 | 27 | 153 | 80 | 233 |
| 38 | 31 | 208 | 109 | 317 |
| 18 | 37 | 196 | 144 | 340 |
| 32 | 51 | 392 | 184 | 576 |
| 20 | 34 | 232 | 121 | 353 |

# Appendix C

**Token Counts Of Pascal Programs**

| $\eta_1$ | $\eta_2$ | $N_1$ | $N_2$ | $N$ |
|---|---|---|---|---|
| 15 | 3 | 30 | 10 | 40 |
| 10 | 4 | 14 | 6 | 20 |
| 7 | 4 | 8 | 5 | 13 |
| 7 | 4 | 8 | 5 | 13 |
| 7 | 4 | 8 | 5 | 13 |
| 11 | 5 | 30 | 22 | 52 |
| 10 | 6 | 15 | 11 | 26 |
| 10 | 6 | 15 | 9 | 24 |
| 12 | 7 | 38 | 32 | 70 |
| 13 | 7 | 21 | 15 | 36 |
| 13 | 7 | 28 | 21 | 49 |
| 14 | 7 | 25 | 17 | 42 |
| 14 | 7 | 25 | 17 | 42 |
| 21 | 7 | 27 | 9 | 36 |
| 9 | 7 | 12 | 7 | 19 |
| 11 | 8 | 17 | 12 | 29 |
| 11 | 8 | 17 | 12 | 29 |
| 11 | 8 | 18 | 13 | 31 |
| 13 | 8 | 24 | 16 | 40 |
| 14 | 8 | 20 | 13 | 33 |
| 14 | 8 | 22 | 14 | 36 |
| 14 | 8 | 25 | 16 | 41 |
| 14 | 8 | 30 | 24 | 54 |
| 16 | 8 | 34 | 24 | 58 |
| 18 | 8 | 31 | 23 | 54 |
| 20 | 8 | 33 | 23 | 56 |
| 24 | 8 | 61 | 32 | 93 |
| 29 | 8 | 86 | 36 | 122 |
| 8 | 8 | 15 | 11 | 26 |
| 10 | 9 | 20 | 14 | 34 |
| 10 | 9 | 33 | 28 | 61 |
| 10 | 9 | 43 | 28 | 71 |
| 11 | 9 | 17 | 12 | 29 |
| 12 | 9 | 22 | 16 | 38 |
| 12 | 9 | 23 | 17 | 40 |
| 13 | 9 | 21 | 15 | 36 |
| 13 | 9 | 21 | 15 | 36 |
| 13 | 9 | 35 | 25 | 60 |
| 13 | 9 | 35 | 25 | 60 |
| 13 | 9 | 35 | 25 | 60 |
| 14 | 9 | 25 | 16 | 41 |
| 14 | 9 | 28 | 18 | 46 |
| 16 | 9 | 30 | 18 | 48 |
| 16 | 9 | 41 | 29 | 70 |
| 18 | 9 | 26 | 24 | 50 |
| 18 | 9 | 36 | 24 | 60 |
| 20 | 9 | 54 | 36 | 90 |
| 26 | 9 | 35 | 21 | 56 |
| 11 | 10 | 16 | 13 | 29 |

| | | | | |
|---|---|---|---|---|
| 11 | 10 | 17 | 13 | 30 |
| 12 | 10 | 25 | 18 | 43 |
| 13 | 10 | 25 | 19 | 44 |
| 14 | 10 | 30 | 18 | 48 |
| 14 | 10 | 30 | 18 | 48 |
| 14 | 10 | 27 | 19 | 46 |
| 14 | 10 | 38 | 27 | 65 |
| 15 | 10 | 29 | 22 | 51 |
| 15 | 10 | 35 | 24 | 59 |
| 15 | 10 | 39 | 33 | 72 |
| 16 | 10 | 35 | 26 | 61 |
| 16 | 10 | 35 | 26 | 61 |
| 16 | 10 | 35 | 26 | 61 |
| 17 | 10 | 45 | 27 | 72 |
| 17 | 10 | 41 | 31 | 72 |
| 18 | 10 | 55 | 44 | 99 |
| 21 | 10 | 52 | 45 | 97 |
| 10 | 11 | 24 | 20 | 44 |
| 11 | 11 | 21 | 17 | 38 |
| 12 | 11 | 24 | 19 | 43 |
| 13 | 11 | 33 | 20 | 53 |
| 13 | 11 | 29 | 25 | 54 |
| 13 | 11 | 36 | 27 | 63 |
| 13 | 11 | 36 | 27 | 63 |
| 13 | 11 | 36 | 27 | 63 |
| 13 | 11 | 36 | 27 | 63 |
| 14 | 11 | 26 | 19 | 45 |
| 14 | 11 | 32 | 24 | 56 |
| 16 | 11 | 33 | 21 | 54 |
| 17 | 11 | 35 | 27 | 62 |
| 18 | 11 | 40 | 30 | 70 |
| 10 | 12 | 27 | 22 | 49 |
| 12 | 12 | 25 | 26 | 51 |
| 12 | 12 | 38 | 27 | 65 |
| 13 | 12 | 30 | 20 | 50 |
| 13 | 12 | 30 | 20 | 50 |
| 13 | 12 | 28 | 21 | 49 |
| 14 | 12 | 30 | 23 | 53 |
| 14 | 12 | 39 | 29 | 68 |
| 15 | 12 | 26 | 17 | 43 |
| 15 | 12 | 29 | 19 | 48 |
| 15 | 12 | 54 | 33 | 87 |
| 16 | 12 | 33 | 22 | 55 |
| 16 | 12 | 33 | 22 | 55 |
| 16 | 12 | 33 | 22 | 55 |
| 16 | 12 | 36 | 28 | 64 |
| 17 | 12 | 41 | 27 | 68 |
| 18 | 12 | 40 | 33 | 73 |
| 19 | 12 | 40 | 28 | 68 |
| 19 | 12 | 44 | 30 | 74 |
| 19 | 12 | 63 | 34 | 97 |

| | | | | |
|---|---|---|---|---|
| 19 | 12 | 54 | 41 | 95 |
| 19 | 12 | 54 | 41 | 95 |
| 21 | 12 | 28 | 20 | 48 |
| 37 | 12 | 83 | 32 | 115 |
| 12 | 13 | 19 | 15 | 34 |
| 13 | 13 | 25 | 18 | 43 |
| 13 | 13 | 35 | 24 | 59 |
| 13 | 13 | 33 | 27 | 60 |
| 13 | 13 | 52 | 32 | 84 |
| 15 | 13 | 38 | 27 | 65 |
| 15 | 13 | 40 | 33 | 73 |
| 16 | 13 | 63 | 46 | 109 |
| 17 | 13 | 42 | 24 | 66 |
| 17 | 13 | 63 | 46 | 109 |
| 17 | 13 | 63 | 46 | 109 |
| 17 | 13 | 63 | 46 | 109 |
| 18 | 13 | 40 | 30 | 70 |
| 18 | 13 | 51 | 50 | 101 |
| 19 | 13 | 36 | 20 | 56 |
| 19 | 13 | 51 | 34 | 85 |
| 20 | 13 | 47 | 29 | 76 |
| 22 | 13 | 51 | 32 | 83 |
| 27 | 13 | 58 | 44 | 102 |
| 11 | 14 | 17 | 14 | 31 |
| 11 | 14 | 38 | 28 | 66 |
| 14 | 14 | 36 | 26 | 62 |
| 14 | 14 | 55 | 48 | 103 |
| 15 | 14 | 32 | 26 | 58 |
| 15 | 14 | 45 | 36 | 81 |
| 15 | 14 | 45 | 36 | 81 |
| 15 | 14 | 45 | 36 | 81 |
| 18 | 14 | 45 | 33 | 78 |
| 19 | 14 | 57 | 44 | 101 |
| 20 | 14 | 52 | 35 | 87 |
| 20 | 14 | 89 | 74 | 163 |
| 21 | 14 | 53 | 33 | 86 |
| 10 | 15 | 24 | 19 | 43 |
| 10 | 15 | 24 | 19 | 43 |
| 11 | 15 | 39 | 26 | 65 |
| 12 | 15 | 29 | 22 | 51 |
| 12 | 15 | 32 | 27 | 59 |
| 14 | 15 | 29 | 22 | 51 |
| 14 | 15 | 36 | 28 | 64 |
| 15 | 15 | 42 | 32 | 74 |
| 15 | 15 | 51 | 36 | 87 |
| 16 | 15 | 38 | 29 | 67 |
| 17 | 15 | 39 | 25 | 64 |
| 17 | 15 | 51 | 37 | 88 |
| 17 | 15 | 59 | 49 | 108 |
| 18 | 15 | 154 | 40 | 194 |
| 40 | 15 | 138 | 70 | 208 |

| | | | | |
|---|---|---|---|---|
| 11 | 16 | 26 | 20 | 46 |
| 12 | 16 | 31 | 28 | 59 |
| 12 | 16 | 42 | 30 | 72 |
| 12 | 16 | 42 | 30 | 72 |
| 12 | 16 | 42 | 30 | 72 |
| 12 | 16 | 42 | 30 | 72 |
| 12 | 16 | 51 | 46 | 97 |
| 13 | 16 | 35 | 26 | 61 |
| 15 | 16 | 48 | 38 | 86 |
| 16 | 16 | 50 | 38 | 88 |
| 17 | 16 | 47 | 36 | 83 |
| 18 | 16 | 68 | 55 | 123 |
| 20 | 16 | 48 | 32 | 80 |
| 20 | 16 | 52 | 39 | 91 |
| 20 | 16 | 52 | 39 | 91 |
| 21 | 16 | 72 | 52 | 124 |
| 21 | 16 | 72 | 52 | 124 |
| 21 | 16 | 72 | 52 | 124 |
| 22 | 16 | 51 | 33 | 84 |
| 23 | 16 | 74 | 50 | 124 |
| 26 | 16 | 82 | 66 | 148 |
| 27 | 16 | 82 | 69 | 151 |
| 38 | 16 | 121 | 68 | 189 |
| 5 | 16 | 53 | 37 | 90 |
| 12 | 17 | 25 | 19 | 44 |
| 12 | 17 | 32 | 24 | 56 |
| 12 | 17 | 32 | 24 | 56 |
| 12 | 17 | 37 | 28 | 65 |
| 12 | 17 | 46 | 34 | 80 |
| 18 | 17 | 60 | 51 | 111 |
| 21 | 17 | 53 | 39 | 92 |
| 21 | 17 | 59 | 40 | 99 |
| 22 | 17 | 64 | 48 | 112 |
| 30 | 17 | 176 | 151 | 327 |
| 32 | 17 | 55 | 35 | 90 |
| 12 | 18 | 32 | 26 | 58 |
| 12 | 18 | 42 | 32 | 74 |
| 13 | 18 | 40 | 32 | 72 |
| 15 | 18 | 40 | 36 | 76 |
| 17 | 18 | 43 | 33 | 76 |
| 19 | 18 | 43 | 31 | 74 |
| 20 | 18 | 68 | 48 | 116 |
| 20 | 18 | 90 | 73 | 163 |
| 22 | 18 | 57 | 39 | 96 |
| 22 | 18 | 61 | 45 | 106 |
| 25 | 18 | 68 | 52 | 120 |
| 27 | 18 | 73 | 66 | 139 |
| 28 | 18 | 63 | 44 | 107 |
| 14 | 19 | 54 | 37 | 91 |
| 14 | 19 | 39 | 41 | 80 |
| 16 | 19 | 52 | 40 | 92 |

| | | | | |
|---|---|---|---|---|
| 19 | 19 | 66 | 47 | 113 |
| 19 | 19 | 72 | 50 | 122 |
| 24 | 19 | 77 | 52 | 129 |
| 35 | 19 | 93 | 55 | 148 |
| 11 | 20 | 32 | 26 | 58 |
| 15 | 20 | 51 | 41 | 92 |
| 16 | 20 | 51 | 41 | 92 |
| 16 | 20 | 57 | 48 | 105 |
| 16 | 20 | 75 | 60 | 135 |
| 17 | 20 | 50 | 41 | 91 |
| 18 | 20 | 51 | 36 | 87 |
| 18 | 20 | 59 | 47 | 106 |
| 19 | 20 | 59 | 38 | 97 |
| 21 | 20 | 80 | 59 | 139 |
| 22 | 20 | 116 | 96 | 212 |
| 22 | 20 | 116 | 96 | 212 |
| 23 | 20 | 62 | 45 | 107 |
| 23 | 20 | 73 | 52 | 125 |
| 24 | 20 | 116 | 85 | 201 |
| 26 | 20 | 69 | 48 | 117 |
| 31 | 20 | 126 | 106 | 232 |
| 44 | 20 | 269 | 167 | 436 |
| 12 | 21 | 39 | 33 | 72 |
| 17 | 21 | 45 | 33 | 78 |
| 17 | 21 | 55 | 58 | 113 |
| 20 | 21 | 62 | 46 | 108 |
| 25 | 21 | 100 | 76 | 176 |
| 29 | 21 | 123 | 101 | 224 |
| 7 | 21 | 55 | 45 | 100 |
| 13 | 22 | 49 | 38 | 87 |
| 16 | 22 | 33 | 22 | 55 |
| 17 | 22 | 44 | 32 | 76 |
| 20 | 22 | 76 | 60 | 136 |
| 22 | 22 | 89 | 58 | 147 |
| 24 | 22 | 55 | 40 | 95 |
| 24 | 22 | 76 | 53 | 129 |
| 25 | 22 | 115 | 81 | 196 |
| 26 | 22 | 133 | 103 | 236 |
| 28 | 22 | 133 | 103 | 236 |
| 35 | 22 | 169 | 69 | 238 |
| 15 | 23 | 43 | 39 | 82 |
| 16 | 23 | 71 | 60 | 131 |
| 18 | 23 | 60 | 45 | 105 |
| 19 | 23 | 74 | 57 | 131 |
| 21 | 23 | 87 | 23 | 110 |
| 21 | 23 | 96 | 81 | 177 |
| 26 | 23 | 111 | 73 | 184 |
| 27 | 23 | 71 | 41 | 112 |
| 43 | 23 | 304 | 216 | 520 |
| 11 | 24 | 48 | 41 | 89 |
| 18 | 24 | 86 | 62 | 148 |

| | | | | |
|---|---|---|---|---|
| 18 | 24 | 79 | 65 | 144 |
| 18 | 24 | 89 | 77 | 166 |
| 21 | 24 | 94 | 72 | 166 |
| 23 | 24 | 80 | 61 | 141 |
| 24 | 24 | 96 | 87 | 183 |
| 29 | 24 | 90 | 67 | 157 |
| 29 | 24 | 94 | 67 | 161 |
| 13 | 25 | 50 | 39 | 89 |
| 13 | 25 | 59 | 53 | 112 |
| 14 | 25 | 54 | 43 | 97 |
| 15 | 25 | 58 | 43 | 101 |
| 15 | 25 | 78 | 64 | 142 |
| 17 | 25 | 77 | 62 | 139 |
| 18 | 25 | 18 | 72 | 90 |
| 21 | 25 | 76 | 55 | 131 |
| 25 | 25 | 102 | 57 | 159 |
| 26 | 25 | 117 | 88 | 205 |
| 29 | 25 | 155 | 114 | 269 |
| 30 | 25 | 160 | 114 | 274 |
| 13 | 26 | 59 | 35 | 94 |
| 19 | 26 | 45 | 29 | 74 |
| 26 | 26 | 90 | 69 | 159 |
| 28 | 26 | 89 | 69 | 158 |
| 17 | 27 | 53 | 40 | 93 |
| 19 | 27 | 80 | 67 | 147 |
| 20 | 27 | 90 | 73 | 163 |
| 27 | 27 | 119 | 69 | 188 |
| 37 | 27 | 90 | 57 | 147 |
| 41 | 27 | 163 | 111 | 274 |
| 33 | 28 | 119 | 96 | 215 |
| 26 | 29 | 125 | 108 | 233 |
| 31 | 29 | 117 | 100 | 217 |
| 33 | 29 | 124 | 98 | 222 |
| 39 | 29 | 114 | 161 | 275 |
| 21 | 30 | 122 | 91 | 213 |
| 28 | 30 | 81 | 67 | 148 |
| 31 | 30 | 196 | 124 | 320 |
| 17 | 31 | 66 | 51 | 117 |
| 20 | 31 | 91 | 70 | 161 |
| 20 | 31 | 94 | 77 | 171 |
| 22 | 31 | 149 | 120 | 269 |
| 22 | 32 | 159 | 132 | 291 |
| 24 | 32 | 161 | 133 | 294 |
| 48 | 32 | 370 | 249 | 619 |
| 17 | 33 | 96 | 80 | 176 |
| 20 | 35 | 107 | 80 | 187 |
| 22 | 35 | 122 | 104 | 226 |
| 25 | 35 | 89 | 82 | 171 |
| 29 | 35 | 102 | 63 | 165 |
| 22 | 37 | 81 | 63 | 144 |
| 28 | 37 | 132 | 93 | 225 |

| | | | | |
|---|---|---|---|---|
| 27 | 38 | 161 | 131 | 292 |
| 32 | 38 | 308 | 132 | 440 |
| 27 | 39 | 113 | 90 | 203 |
| 26 | 40 | 153 | 125 | 278 |
| 21 | 41 | 86 | 76 | 162 |
| 21 | 41 | 88 | 77 | 165 |
| 27 | 42 | 119 | 116 | 235 |
| 28 | 42 | 169 | 123 | 292 |
| 25 | 43 | 135 | 116 | 251 |
| 29 | 46 | 206 | 127 | 333 |
| 43 | 50 | 227 | 130 | 357 |
| 21 | 51 | 312 | 296 | 608 |
| 32 | 54 | 205 | 130 | 335 |
| 17 | 55 | 148 | 129 | 277 |
| 22 | 56 | 230 | 204 | 434 |
| 23 | 59 | 371 | 296 | 667 |
| 35 | 66 | 316 | 178 | 494 |
| 43 | 74 | 594 | 395 | 989 |
| 39 | 101 | 414 | 308 | 722 |
| 45 | 111 | 770 | 576 | 1346 |
| 42 | 116 | 634 | 496 | 1130 |
| 39 | 145 | 314 | 104 | 418 |
| 37 | 147 | 398 | 243 | 641 |
| 40 | 151 | 714 | 560 | 1274 |
| 39 | 152 | 411 | 257 | 668 |
| 43 | 154 | 475 | 304 | 779 |
| 44 | 155 | 454 | 280 | 734 |
| 42 | 159 | 455 | 291 | 746 |
| 44 | 164 | 547 | 358 | 905 |
| 48 | 168 | 474 | 301 | 775 |
| 38 | 169 | 693 | 527 | 1220 |
| 41 | 172 | 805 | 624 | 1429 |
| 45 | 179 | 709 | 483 | 1192 |
| 47 | 186 | 1057 | 770 | 1827 |
| 45 | 195 | 568 | 388 | 956 |
| 46 | 195 | 880 | 595 | 1475 |
| 44 | 202 | 1350 | 975 | 2325 |
| 51 | 203 | 705 | 470 | 1175 |
| 48 | 209 | 766 | 524 | 1290 |
| 50 | 214 | 794 | 579 | 1373 |
| 54 | 232 | 962 | 770 | 1732 |
| 39 | 243 | 666 | 418 | 1084 |
| 50 | 247 | 1223 | 957 | 2180 |
| 40 | 251 | 680 | 436 | 1116 |
| 37 | 253 | 746 | 489 | 1235 |
| 37 | 262 | 768 | 505 | 1273 |
| 42 | 265 | 769 | 484 | 1253 |
| 41 | 268 | 767 | 488 | 1255 |
| 44 | 268 | 757 | 492 | 1249 |
| 59 | 269 | 578 | 499 | 1077 |
| 42 | 271 | 759 | 477 | 1236 |

| | | | | |
|---|---|---|---|---|
| 51 | 271 | 1113 | 852 | 1965 |
| 47 | 275 | 796 | 511 | 1307 |
| 42 | 280 | 784 | 505 | 1289 |
| 56 | 282 | 1356 | 968 | 2324 |
| 46 | 283 | 818 | 593 | 1411 |
| 44 | 285 | 831 | 533 | 1364 |
| 46 | 285 | 805 | 515 | 1320 |
| 44 | 286 | 1295 | 1010 | 2305 |
| 46 | 286 | 638 | 538 | 1176 |
| 48 | 287 | 862 | 540 | 1402 |
| 44 | 290 | 708 | 527 | 1235 |
| 57 | 293 | 1423 | 1019 | 2442 |
| 57 | 303 | 1356 | 955 | 2311 |
| 46 | 310 | 1009 | 696 | 1705 |
| 46 | 313 | 939 | 614 | 1553 |
| 56 | 314 | 1208 | 801 | 2009 |
| 57 | 333 | 1417 | 1043 | 2460 |
| 56 | 339 | 2366 | 1854 | 4220 |
| 50 | 347 | 1663 | 1167 | 2830 |
| 57 | 353 | 2869 | 1947 | 4816 |
| 34 | 367 | 1735 | 1334 | 3069 |
| 56 | 393 | 2720 | 2047 | 4767 |
| 55 | 403 | 1767 | 3988 | 5755 |
| 56 | 405 | 2827 | 2094 | 4921 |
| 60 | 421 | 2177 | 1632 | 3809 |
| 59 | 434 | 5038 | 4073 | 9111 |
| 54 | 462 | 5829 | 4582 | 10411 |
| 52 | 471 | 3497 | 3118 | 6615 |
| 55 | 477 | 2416 | 1660 | 4076 |
| 46 | 510 | 1680 | 1342 | 3022 |
| 55 | 535 | 5855 | 4475 | 10330 |
| 55 | 550 | 5784 | 4428 | 10212 |
| 52 | 599 | 2063 | 1543 | 3606 |
| 56 | 611 | 4385 | 3327 | 7712 |
| 55 | 626 | 5856 | 4480 | 10336 |
| 57 | 632 | 6603 | 5160 | 11763 |
| 58 | 636 | 6099 | 4938 | 11037 |
| 47 | 650 | 5450 | 5498 | 10948 |
| 64 | 686 | 4962 | 3713 | 8675 |
| 54 | 789 | 4396 | 3516 | 7912 |
| 51 | 802 | 4537 | 3662 | 8199 |
| 53 | 805 | 5428 | 4229 | 9657 |
| 60 | 811 | 3402 | 2769 | 6171 |
| 59 | 989 | 10625 | 7946 | 18571 |
| 55 | 1115 | 10421 | 8205 | 18626 |
| 51 | 1143 | 9082 | 6966 | 16048 |
| 53 | 1192 | 7050 | 5544 | 12594 |
| 54 | 1198 | 7571 | 6189 | 13760 |
| 61 | 1393 | 9442 | 7528 | 16970 |

# Appendix D

**Modules of Counting Tokens Of C Language Files**

## Description of Modules

Four modules are designed to implement the work of counting and distinguishing the tokens of the programs written in C language. These four modules should be used in the form of the following script:

% p1 < ProgramName | p2 | sort | p3 | p4 >> ResultsFileName

Every script will produce a line of result appending to the "ResultsFileName". After processing a series of above script, with different "ProgramName", the results are all recorded in the "ResultsFileName" that serves as the data set for the further analysis.

**Module-1** was designed to retrieve all the tokens (or pieces) and comments from the input program. The format of output is simply line by line, each line presents a single token or comment symbol, so that can be processed for the module-2.

**Module-2** was designed to screen out the comment string from the list, and merge some pieces which should not be separated to present a token. For example, in the preprocess section of the program, "#" and "include" should be merge together to be as "#include" to represent a single token. This module was also marking the symbols for particular tokens so that can be easily recognized and classified in the module-4. For example, the tokens which is followed by parenthesis are marked a "*" that means this token is of operator. For the other example, any tokens or strings were quoted by quotation marks were labeled "#" to represent this token is of operand. The output should be sorted before being used in the module-3.

**Module-3** was designed to count the amount of identical tokens, or strings, the output presents the number of occurrence and corresponding token (or string) by lines. The output is used directly to the module-4.

**Module-4** was designed to classify the tokens into operators or operands. The file "keywords" was referred as a library, any token is in this list will be viewed as an operator. Any token with "*" as last character is of operator, with "#" is of operand. All constant numbers, of forms of decimal, hex or oct, are treated as operands. The count of distinct operators and operands, and total number of operators and operands are in the output.

## Module 1

```
program ctoken(input,output);
 type stringtype= array[1..80] of char;
 var  c: char; getastring:boolean; i,k: integer;
      stringvar: stringtype;

 function getchar:char;
  var x:char;
  begin read(x);
        getchar:= x
  end;

 function alph(ch:char):boolean;
  begin
    if (ch in ['a'..'z']) or (ch in ['A'..'Z']) or (ch='_') then
           alph := true
    else alph:= false
  end;
 function alphnum(ch:char):boolean;
  begin
    if (alph(ch)) or (ch in ['0'..'9']) then alphnum:=true
    else alphnum:=false
  end;

 procedure iscomment;
 var done:boolean;
  begin
   done := false; c:=getchar;
     while ( not done ) do
     begin  if (c='*') then begin c:=getchar; if(c='/') then done:=true end
           else c:=getchar
     end;
   write('* this comment */')
  end;

function isblank(st:stringtype;i:integer):boolean;
  var b:boolean; k:integer;
  begin
    b := true;
    for k:=1 to i do
       begin if(st[k]<>' ') then b:=false end;
    isblank := b
  end; (* of isblank function *)

procedure goahead;
  begin
    write(c); c:=getchar
  end;
```

```
procedure formatwrite(cr:char);
  begin
    repeat
      goahead;
    until (c in ['d','u','o','x','X','f','e','E','g','R','G','c','s','%'])
            or (c=cr) ;
    if(c in ['d','u','o','x','X','f','e','E','g','R','G','c','s','%'])
        then goahead
  end;

(* main program *)

begin
  c := getchar;
  while (not eof) do
  begin
  if (c=' ') then c:=getchar
  else if (c='#') then begin goahead; writeln end
  else if (alph(c)) then
     begin
     goahead;
     while (alphnum(c)) do goahead;
     writeln
     end
  else if(c='/') then
      begin goahead;
            if (c='*') then begin iscomment; c:=getchar end;
            if (c='=') then goahead;
            writeln
      end
  else if(c='!') then
      begin goahead;
            if(c='=') then goahead;
            writeln
      end
  else if(c='%')then
      begin goahead;
            if(c='=') then goahead;
            writeln
      end
  else if(c='&')then
      begin goahead;
            if(c='&') or (c='=') then goahead;
            writeln
      end
  else if(c='(')then begin goahead; writeln end
  else if(c=')')then c:=getchar
  else if(c='*')then
      begin goahead;
            if(c='=') then goahead;
            writeln
```

```
   end
else if(c='+')then
    begin goahead;
          if(c='+') or (c='=') then goahead;
          writeln
    end
else if(c=',')then begin goahead; writeln end
else if(c='-')then
    begin goahead;
          if(c='-')or(c='=')or(c='>')then goahead;
          writeln
    end
else if(c=':') then begin goahead; writeln end
else if(c=';') then begin goahead; writeln end
else if(c='<') then
    begin goahead;
          if(c='<') then
             begin goahead;if(c='=') then goahead end;
          if(c='=') then goahead;
          writeln
    end
else if(c='=') then
     begin goahead;
           if(c='=') then goahead;
           writeln
     end
else if(c='>') then
    begin goahead;
          if(c='>') then
             begin goahead;if(c='=') then goahead end;
          if(c='=') then goahead;
          writeln
    end
else if(c='?')then begin goahead; writeln end
else if(c='[')then begin goahead; writeln end
else if(c=']')then c:=getchar
else if(c='{')then begin goahead; writeln end
else if(c='}')then c:=getchar
else if(c='~')then begin goahead; writeln end
else if(c='^')then
    begin  goahead;
          if(c='=') then goahead;
          writeln
    end
else if(c='|')then
     begin goahead;
           if(c='=')or(c='|')then goahead;
           writeln
     end
else if(c='"')then
    begin goahead; writeln;
```

```
        getastring := false;
        i := 1;
        while(c<>'"') do
           begin
              if (c='%')then begin formatwrite('"'); writeln end
              else if (c='`')then
                 begin goahead;
                       if(c in ['0'..'9']) then
                          repeat goahead until (c < '0') or (c > '9')
                       else goahead; writeln
                 end
              else begin getastring:=true;
                       if (i<80) then stringvar[i]:=c;
                       i := i + 1;
                       c:=getchar end
           end;
           i := i-1;
   if(getastring) then begin
     if (isblank(stringvar,i)) then begin
        writeln('blank-string',i:2) end
     else begin
           if i>70 then i:=70;
           for k:=1 to i do write(stringvar[k]);  write('#');
           writeln; end;
   end;
   writeln('#"');
   c:=getchar
   end
else if(c='''')then
  begin goahead; writeln;
        getastring := false;
        i := 1;
        while(c<>'''') do
           begin
              if (c='%')then begin formatwrite('''');writeln end
              else if (c='`')then
                begin goahead;
                       if(c in ['0'..'9']) then
                          repeat goahead until (c < '0') or (c > '9')
                       else goahead;   writeln
                end
              else begin getastring:=true;
                       if(i<80) then stringvar[i]:=c;
                       i:=i+1;
                       c:=getchar end
           end;
           i := i-1;
    if(getastring) then begin
      if(isblank(stringvar,i))then begin
        writeln('blank-string',i:2) end
      else begin
```

```
            if i>70 then i:=70;
            for k:=1 to i do write(stringvar[k]); write('#');
            writeln; end;
        end;
     writeln('#"');
     c :=getchar
    end
else if(c='.') then
      begin goahead;
            if(c in ['0'..'9']) then
               begin
                 goahead;
                 while (c in ['0'..'9']) do goahead;
                 if (c='E') or (c='e') then
                    begin goahead; goahead;
                          while (c in ['0'..'9']) do goahead;
                          if (c='L') or (c='l') then goahead;
                    end
               end;
            writeln
     end
 else if(c='0') then
    begin goahead;
        if (c in ['x','X'])or(c in ['0'..'7'])then
         begin
          if (c = 'X') or (c= 'x') then
            begin
              goahead;
              while (c in ['0'..'9']) or (c in ['a'..'f']) or
                    (c in ['A'..'F']) do goahead;
             if (c='L') or (c='l') then goahead
            end
          else begin
               goahead;
               while(c in ['0'..'7']) do goahead;
               if (c='L') or (c='l') then goahead
               end;
          writeln
         end
        else if(c<>'.') then writeln
        else
    end
  else if(c in ['1'..'9']) then
     begin goahead;
      if(c in ['0'..'9']) or (c='.') then
        begin
          goahead;
          while (c in ['0'..'9']) do goahead;
          if (c='.') then
               begin goahead;
                     if(c in ['0'..'9']) then
```

```
                    begin goahead;
                         while (c in ['0'..'9']) do goahead;
                         if (c='E') or (c='e') then
                         begin goahead; goahead;
                           while (c in ['0'..'9']) do goahead;
                          if (c='L') or (c='l') then goahead;
                        end
                   end
             end;
          if (c='E') or (c='e') then
            begin goahead; goahead;
                 while (c in ['0'..'9']) do goahead;
                     if (c='L') or (c='l') then goahead;
            end
            end;
       if (c='L') or (c='l') then goahead;
       writeln
     end
   else c:=getchar
 end;
 writeln('}the end')
end.
```

## Module 2

```
program p2(input,output);
 type stringtype = record
                content : array[1..80]of char;
                count   : integer
                end;
var s: stringtype;
    n : integer;

procedure get(var s: stringtype);
 var c:char; i:integer;
 begin
   i:= 1;
   repeat
     begin read(c);  s.content[i]:=c; i := i + 1 end
   until (eoln) or (eof);
   s.count:= i-1; read(c)
 end;

procedure put(s: stringtype);
 var i:integer;
 begin for i:= 1 to s.count do write(s.content[i]) end;


function verify(s:stringtype):integer;
 (* EOF=0; identifier=1; preprocessor=2; comment=3; equal=5; else=4 *)
 var c,last :char;
 begin
   c:=s.content[1]; verify:=4; last := s.content[s.count];
   if (c='}') and (last<>'#') and
                (s.count=8) and (s.content[5]=' ') then verify:=0;
   if((c in ['a'..'z'])or(c in ['A'..'Z'])or(c ='_'))  and
       (last <> '#') then verify:=1;
   if(c ='#') and (s.count=1) then verify:=2;
   if(c='#') and (s.count=2) and (s.content[s.count]='"') then
       verify:=3;
   if(c='/') and (s.count>10) and (last<>'#') then verify:=3;
   if(c='=') and (s.count=1) then verify:=5;
 end;

begin
 get(s); n := verify(s);
 while(n <> 0) do
   begin
    if(n=3) then get(s)
    else if(n=1) then
      begin
        put(s); get(s);
        if(s.content[1]='(') and (s.count=1) then write('*');
```

```
        writeln
      end
    else if(n=2)then (* preprocessor # *)
      begin
        put(s); get(s);
        if (s.content[1] in ['a'..'z']) or
           (s.content[1] in ['A'..'Z']) or (s.content[1]='_') then
           begin
                put(s); writeln; get(s);
                   if(s.content[1] = '<') then
                     begin
                        put(s);write('>'); writeln; get(s);
                        while(s.content[1]<>'>')do begin put(s); get(s) end;
                      writeln;
                      get(s)
                     end
           end
        else writeln;
      end
    else if (n=5) then
      begin
        put(s); writeln; get(s);
        if (s.count=1) then
           begin if (s.content[1]='''') or (s.content[1]='"') then
                begin put(s); writeln; get(s);
                      while(s.count<>2) or (s.content[1]<>'#') or
                           (s.content[2]<>'"') do begin
                           put(s); get(s)  end;
                      write ('#');
                      writeln
                end
           end
      end
    else begin put(s); get(s); writeln end;
    n:=verify(s)
   end
end.
```

## Module 3

```
program p3(input,output);
 type stringtype = record
                content : array[1..80]of char;
                count   : integer
                end;
 var s1,s2: stringtype; n:integer;

procedure get(var s: stringtype);
 var c:char; i:integer;
 begin
   i:= 1;
   repeat
     begin read(c);  s.content[i]:=c; i := i + 1 end
   until (eoln) or (eof);
   s.count:= i-1; read(c)
 end;

procedure put(s: stringtype);
 var i:integer;
 begin for i:= 1 to s.count do write(s.content[i]) end;

function compare(s1,s2:stringtype):boolean;
 var i:integer;
 begin
   if(s1.count=s2.count)then
      begin
        compare:=true;
        for i:=1 to s1.count do
           if(s1.content[i]<>s2.content[i])then compare:=false
      end
    else compare:=false
 end;

begin
 n:=1;
 get(s1);
 while (not eof ) do
     begin
     get(s2);
     if(compare(s1,s2))then n:=n+1
     else begin  write(n:5,'   '); put(s1);writeln; s1:=s2; n:=1 end;
     if(eof) then begin write(n:5,'   '); put(s1);writeln; s1:=s2; n:=1 end;
   end
end.
```

## Module 4

```
program p4(input,output);
  const nkey=33;
  type stringtype = record
                 content : array[1..80]of char;
                 count   : integer
                 end;
var
 keyfile : text;
 s : stringtype;
 key: array[1..nkey]of stringtype;
 i, r, m, d, dn, n, nrd: integer;
 iskey : boolean;
 c: char;

procedure get(var s: stringtype);
 var c:char; i:integer;
 begin
   i:= 1;
   repeat
     begin read(c);  s.content[i]:=c; i := i + 1 end
   until (eoln) or (eof) ;
   s.count:= i-1; read(c)
 end;

function conv(s:stringtype):integer;
 var i, k : integer;
 begin
   k := 0;
   for i:=1 to 5 do
     begin
        if(s.content[i]<>' ') then
          begin
           if(i=5)then k:=k+ ord(s.content[i])-48
           else if(i=4) then k:=k+10*(ord(s.content[i])-48)
           else if(i=3) then k:=k+100*(ord(s.content[i])-48)
           else if(i=2) then k:=k+1000*(ord(s.content[i])-48)
           else k:=k+10000*(ord(s.content[i])-48)
          end
     end;
   conv := k
 end;

procedure getkey(var s: stringtype);
 var c:char; i:integer;
 begin
   i:= 1;
   repeat
     begin read(keyfile,c);  s.content[i]:=c; i := i + 1 end
```

```
    until (eoln(keyfile)) or (eof(keyfile));
    s.count:= i-1; read(keyfile,c)
  end;

function compare(s1,s2:stringtype):boolean;
 var i:integer;
 begin
   if(s1.count=s2.count-8)then
      begin
        compare:=true;
        for i:=1 to s1.count do
           if(s1.content[i]<>s2.content[i+8])then compare:=false
      end
   else compare:=false
 end;

begin
 r:=0;rn:=0;d:=0;dn:=0;n:=0;
 reset(keyfile,'keywords');
 for i:= 1 to nkey do begin getkey(s); key[i]:=s end;
 repeat
   begin
     get(s); c := s.content[9];
     if(c in ['0'..'9']) or (c='') then
        begin d:=d+1; dn:=dn+conv(s) end
     else if (c='.')and(s.count>9) then
        begin d:=d+1; dn:=dn+conv(s) end
     else if (c in ['a'..'z'])or(c in ['A'..'Z'])or(c='_') then
            begin if(s.content[s.count]='*')then
                       begin r:=r+1; rn:=rn+conv(s) end
                  else if (s.content[s.count]='#')then
                       begin d:=d+1; dn:=dn+conv(s) end
                  else begin
                       iskey:=false;
                       for i:=1 to nkey do
                           if(compare(key[i],s)) then iskey:=true;
                        if(iskey)then begin r:=r+1; rn:=rn+conv(s) end
                        else begin  d:=d+1; dn:=dn+conv(s) end
                        end
            end
      else begin if (s.content[s.count]='#')then
                       begin d:=d+1; dn:=dn+conv(s) end
                 else begin  r:=r+1; rn:=rn+conv(s) end
         end
   end;
 until (eof);
 n := dn+rn;
 nrd := d+r;
 writeln(r:8,d:8,nrd:8,rn:8,dn:8,n:8);
end.
```

## The List Of Keywords

auto
break
case
char
continue
default
do
double
else
enum
extern
float
for
goto
if
int
long
register
return
short
sizeof
static
struct
switch
typedef
union
unsigned
void
while
FILE
stdin
stdout
stderr

# AN EMPIRICAL INVESTIGATION
# OF
# HALSTEAD'S SOFTWARE LENGTH FORMULA

by

CHERN-HWANG HWANG

B.A., Tunghai University, 1977
M.S., Kansas State University, 1984

AN ABSTRACT OF A REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1988

# ABSTRACT

This report discusses the existing software length estimation models, which suggested by Halstead, Fitsos, Albrecht and Jensen respectively. Three data sets, UNIX source programs, CMPSC 541 programs, and Pascal programs are used to develop new models and investigate the characteristics of the models. The raw data sets are normalized by logarithmic function, so that the transformed data sets can satisfy the requirements of further statistical modeling procedures. In this report, the author proposes four models which are developed based upon linear and nonlinear modeling methods.

The experiments are conducted to compare the accuracy of the models. All the models present high correlation between the estimated length and the actual length. However, correlation analysis is not sufficient to show the superiority or inferiority among the models; therefore, the mean of relative error (MRE) and the counts of overestimating and underestimating were employed for the further comparisons. The results show the models derived from linear modeling provide more accuracy estimation than any other models do, having not only the lower MRE but also the balancing counts of over- and under- estimating. The Halstead's model tends to overestimate the small programs and underestimate the large ones. Fitsos's and Albercht's models are suitable for large programs but not for the small ones, and Albercht's model is much more accurate than Fitsos's. Jensen's model, with its simple structure, accurately estimate the programs of not large size, but the error increases when the program size grows. The models developed from nonlinear modeling, provide moderate accuracy of estimation.