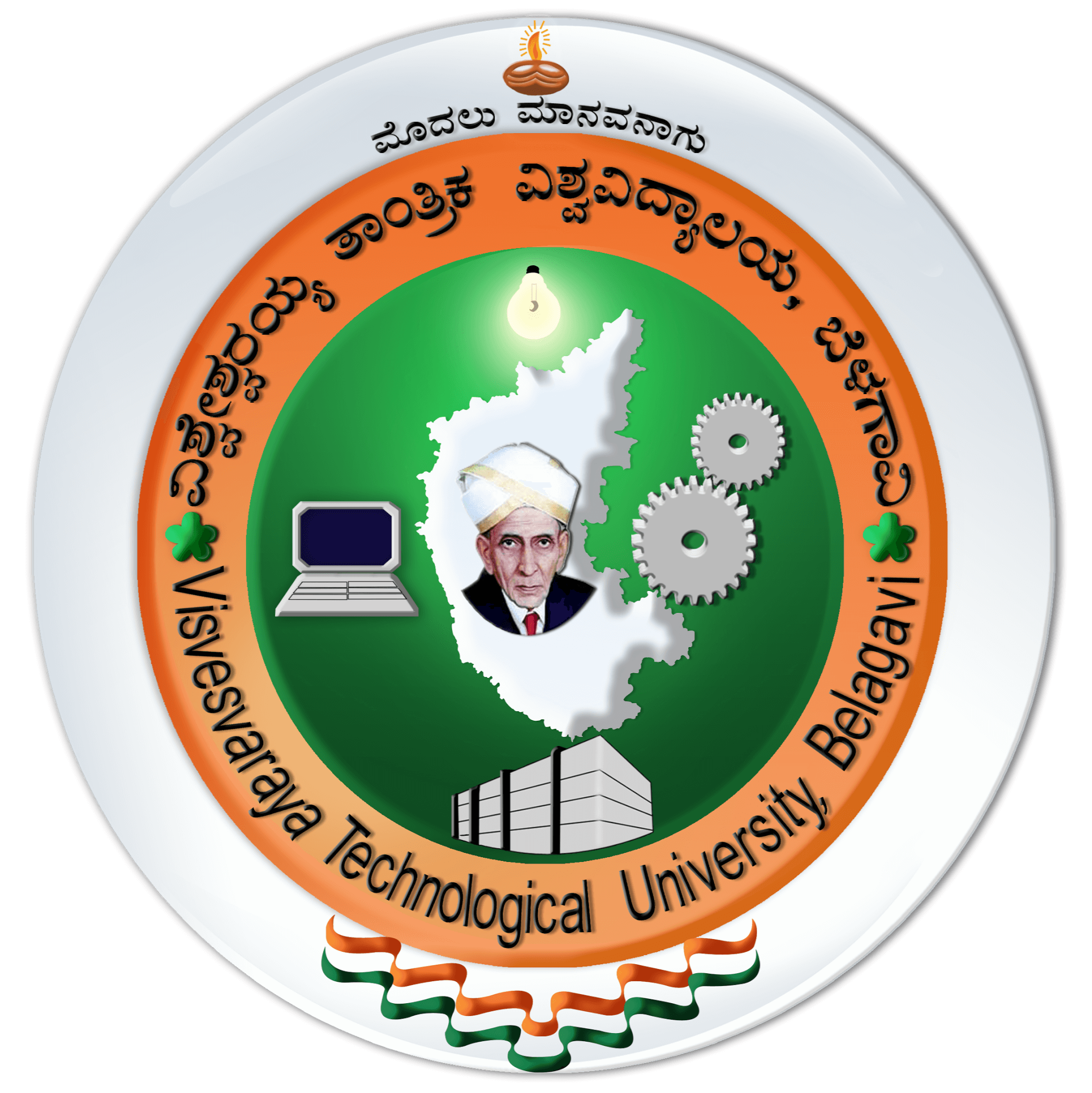
###### VISVESVARAYA TECHNOLOGICAL UNIVERSITY,

**BELGAUM - 590014**



**A Minor Project Report**

**on**

**Programmer’s efficiency analyser**

**submitted in partial fulfilment of the requirement for the degree of,**

**Bachelor of Engineering in**

**Computer Science and Engineering**

**Submitted By,**

|  |  |
| --- | --- |
| Srushti V Patil | 2BV14CS110 |
| Sudeshna Chakraborty | **2BV14CS111** |
| Tejaswini | **2BV14CS122** |
| Venuprasad Naik | **2BV14CS124** |

### 

### Under the guidance of,

**Prof. Mahesh S Patil**



Department of Computer Science and Engineering

B.V.B. COLLEGE OF ENGINEERING & TECHNOLOGY,

**(An Autonomous Institution affiliated to VTU, Belgaum)**

Academic year 2016-17

K.L.E. SOCIETY’S

**B.V. BHOOMARADDI COLLEGE OF ENGINEERING & TECHNOLOGY, HUBLI - 580031**

**(An Autonomous Institution affiliated to VTU, Belgaum)**

2016 - 2017



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

**CERTIFICATE**

This is to certify that Mini Project entitled VIKI is a bonafied work carried out by the student team Ms. SRUSHTI PATIL – USN-2BV14CS110, Ms. SUDESHNA CHAKRABORTY. – USN-2BV14CS111, Ms. TEJASWINI – USN-2BV14CS122, Mr. VENUPRASAD NAIK – USN-2BV14CS124, in partial fulfilment of completion of sixth semester B. E. in Computer science and Engineering during the year 2016 – 2017. The project report has been approved as it satisfies the academic requirement with respect to the project work prescribed for the above said program.

**Guides H.O.D Principal**

**Prof. Mahesh S Patil Dr. G. H. Joshi Dr. P. G. Tewari**

**External examiners:**

**Name of the Examiner Signature with date**

**1.**

**2.**

**ABSTRACT**

Usually the programs which give accurate results is mistaken with efficient code, which is not necessary to be so. Productivity is measured in terms of “LOC / PM (Lines of code per person month)” but again a programmer who can write thousands of lines of code cannot be taken granted to be an efficient programmer. Efficient coders aim at higher productivity with efficient usage of constructs. How to judge which code is more efficient when its results are same, on which parameters can one decide on which program is more efficient is the question. Automated tool can be designed to find programmer’s efficiency, which considers the parameters such as, usage of memory and machine cycles or frequently used constructs. Which results is differentiating between programmer’s who can write codes giving required output, and codes making efficient usage of memory and CPU(machine cycles involved in computations).

Our project is a software tool which takes C code as input and gives user, the percentage of inefficiencies, based on parameters discussed in the above paragraph.

**ACKNOWLEDGEMENT**

We are thankful to our beloved Principal **Dr. P.G. Tewari** for giving us an opportunity to work and for providing the necessary facilities in our college.

We take this opportunity to express our deep sense of gratitude to our H.O.D **Dr. G. H. Joshi,** Department of Computer Science, BVB College of Engineering and Technology for his support. We extend our thanks to our guide **Prof. Mahesh S Patil** for his constant guidance.

We also express our gratitude to our parents and friends for their encouragement. We would also like to thank everyone who helped us directly or indirectly in carrying out this project successfully.

### SRUSHTI V PATIL 2BV14CS110

SUDESHNA CHAKRABORTY 2BV14CS111

TEJASWINI 2BV14CS122

VENUPRASAD NAIK 2BV14CS124

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Chapter No.** | **Table of Contents** | | | | |
| **1.** | **Preamble** | | | | |
|  | 1.1 | Introduction | | | |
|  | 1.2 | Problem definition | | | |
|  | 1.3 | Scope and Objectives of the project | | | |
| **2.** | **Literature Survey** | | | | |
|  | 2.1 | Proposed System | | | |
|  | 2.2 | Advantages/applications of proposed system | | | |
|  | 2.3 | Constraints or limitations of proposed system | | | |
| **3.** | **Software Requirement Specification** | | | | |
|  | 3.1 | Overview of SRS | | | |
|  | 3.2 | Intended audience | | | |
|  | 3.3 | Requirement Specifications | | | |
|  |  | 3.3.1 | | Functional Requirements | |
|  |  | 3.3.2 | | Use case diagrams | |
|  |  |  | | Use descriptions using scenarios, strictly as per Pressman Template | |
|  |  | 3.3.3 | Non-functional Requirements | | |
|  |  |  | | 3.3.3.1 | Portability |
|  |  |  | | 3.3.3.2 | Usability |
|  |  |  | | 3.3.3.3 | Response Time |
|  |  |  | | 3.3.3.4 | Graphical User Interface |
|  | 3.4 | Software requirement specifications | | | |
|  | 3.5 | Acceptance Test-Plan | | | |
| **4** | **System Design** | | | | |
|  | 4.1 | Data Structures used and Justification | | | |
|  | 4.2 | Level 0 DFD | | | |
|  | 4.3 | GUI Wireframe | | | |
|  | 4.4 | Class Diagram | | | |

|  |  |  |
| --- | --- | --- |
|  | 4.5 | Class coupling matrix and explanation |
| **5** | **Implementation** | |
|  | 5.1 | Implementation |
|  | 5.2 | Introduction-Pair programming |
|  | 5.3 | Brief description of modules |
|  | 5.4 | Module Description |
| **6** | **Testing**  6.1 Introduction | |
|  | 6.2 | Module test plan and test cases |
|  | 6.3 | JUNIT |
|  | 6.4 | Usability testing report |
| **7** | **Results** | |
| **8** | **Summary of your learning** | |
| **9** | **Conclusion and future scope** | |
| **10** | **References/Bibliography** | |
| **11** | **Appendix** | |
|  | A | Glossary |
|  | B | Explanation on JUNIT tool |
|  | C | Feedback |
|  | D | Screenshots |

**1. Preamble**

**1.1 Introduction**

“Programmer’s efficiency analyser” is a software tool which reads a C code uploaded by a user or an organisation, parses it using ANTLR and later the results are used to analyse the pattern a programmer used within a constructor and then based on few pre-defined weightage assigned and computations, the programmer is given a graph representing his inefficient constructs along with percentage of inefficiency.

* 1. **Problem definition**
     1. Identification process:

In today’s competitive world, software plays an important role. Performance, ease, portability, compactness are the basic needs of any developed system software. And so is the need to have an efficient programmer, who can use memory and CPU efficiently to produce a product which satisfies the needs of the hour. And hence we need an automated tool which can tell, who is efficient programmer.

* + 1. Problem Description:

when given two programs, one which yields accurate results, and another which is incorrect it’s very easy to tell who is efficient programmer. But, when given with two programs working perfectly fine, how can one judge who is efficient programmer. Our tool is all about solving the problem of identifying the efficiency of a programmer.

* 1. **Scope and Objectives of the project**
     1. Scope:

The present design of the tool is limited to judge a C code, and also there are predefined set of constructs which the tool looks for while parsing through the code, on encountering such constructs analysis for efficient usage of constructs is verified, which is again pre-defined Ex: usage of function calls in conditional expression of for loop is acceptable unless the parameters passed to the function call is being modified by the code, for every iteration. Likewise, for few constructs analysis methodology is defined and is checked for the same in the input code.

* + 1. Objectives:

The main objective behind developing this tool is to help the organizations/educational institutions, So that they can upload the code written by their students and get the results as to who is the most efficient programmer, it also helps the individual to upload his/her own code and get to know where he/she stands in writing efficient code, so that they can improve further.

**2. Literature survey**

**2.2 Proposed System**

“Programmer’s efficiency analyser” is a software tool which is integrated onto the cloud, which can be accessed by anyone anytime, on uploading their code, the code is analysed and the programmer is given with graph representing the scores, corresponding to each inefficient usage of construct. which depicts where the programmer is actually week in usage of the constructs, it may be with respect to memory usage, or improper usage of CPU.

**2.3 Advantages of the proposed System**

2.3.1 Any user can check his efficiency.

2.3.2 Quick output which saves time and energy.

2.3.3 Results will be in two forms inefficiency percentage and bar graph.

2.3.4 Any user can use this software tool without depending on anyone.

**2.4 Constraints or limitations of proposed system**

2.4.1 Internet is required for getting connected to cloud.

2.4.2 Should have a C-program to be given as input.

2.4.3 Users should know about C-coding standards.

**3. Software Requirement Specification**

**3.1 Overview of SRS**

The SRS document explains the requirements for the project including functional, non-functional requirements along with use case diagram and description.

**3.2 Intended Audience**

All end users who are interested in using the Internet with ease.

**3.3 Requirement Specifications:**

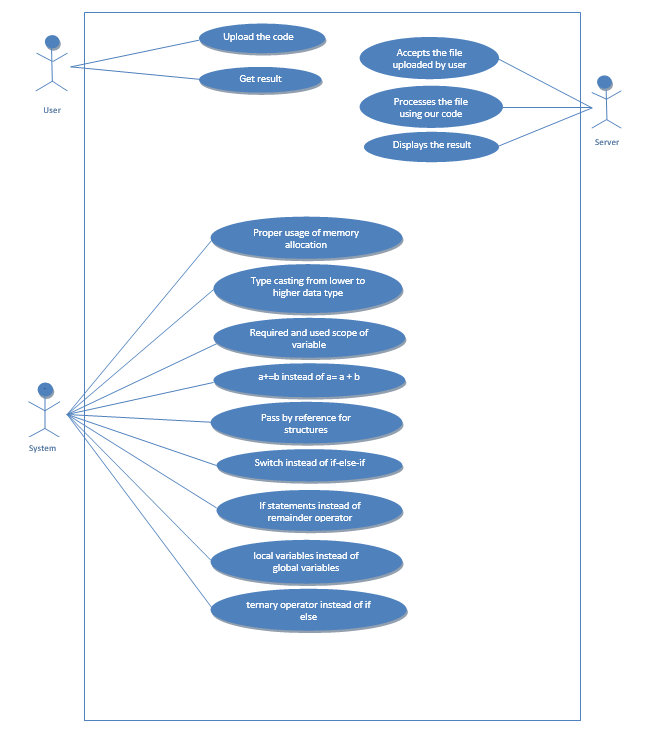
3.3.1 Functional Requirements:

|  |  |  |
| --- | --- | --- |
| **Req.id** | **Functional Requirements** | **Priority** |
| **1.** | User Shall be able to upload the code. | **1** |
| **2.** | User shall be able to get result in form of percentage and graph. | **1** |
| **3.1** | Server shall be able to check if one assignment statement in construct then if else can be replaced by ternary operator. | **2** |
| **3.2** | Server shall be able to check the efficiency of the programmer and display result based on Declared variables and variables used in the program. | **3** |
| **3.3** | Server shall be able to check the efficiency of the programmer and display result based on Type casting. | **3** |
| **3.4** | Server shall be able to check the efficiency of the programmer and display result based on Freeing of allocated memory for pointers. | **1** |
| **3.5** | Server shall be able to check the efficiency of the programmer and display result based on Usage of Switch instead of if-else-if construct. | **2** |
| **3.6** | Server shall be able to check the efficiency of the programmer and display result based on Usage of statements like a+=b instead of a=a + b | **2** |

|  |  |  |
| --- | --- | --- |
| **3.7** | Server shall be able to check the efficiency of the programmer and display result based on If parameters are changing in conditional statements of looping. | **1** |
| **3.8** | Server shall be able to check the efficiency of the programmer and display result based on usage of ternary operator when one assignment statement is present in construct. | **2** |

**Figure 1**

**3.3.2 Use Case Diagrams:**



**Figure 2**

**Use Case Description:**

**Using Figure 2,**

|  |
| --- |
| **1.** **Use case: Upload the code.** |
| Actor: Users  Pre-condition: User is able to access the server.  Post-condition: User will be able to successfully upload the code in the server.  Success Scenario:  1.1 User shall be able to select the program file to be uploaded.  1.2 User shall be able to successfully upload the code to the server.  Exception Scenario:  1.2.a. User may fail to upload the program due to internet connection failure or low internet connection. |

|  |
| --- |
| **2.** **Use case: Get result** |
| Actor: Users  Pre-condition: User is able to upload the code in the server.  Post-condition: User will be able to successfully obtain the result.  Success Scenario:  2.1 User shall be able to successfully obtain the result both in percentage as well as graph form.  Exception Scenario:  2.1.a. User may encounter an error if anything other than a c program is uploaded.  2.1.b. User may encounter an error if the internet connection fails. |

|  |
| --- |
| **3.** **Use case: Required and used scope of variables** |
| Actor: System  Pre-condition: User has uploaded the code.  Post-condition: System will be able to determine whether the code is efficient.  Success Scenario:  3.1 System shall be able to analyse the code.  3.2 System shall be able to compare between the required scope of variables and the scope used in program.  3.3 System shall be able to successfully deliver the result.  Exception Scenario:  3.3.a System may fail to deliver the result if internet connection is low or failed. |
| **4.** **Use case: Proper usage of memory allocation** |
| Actor: System  Pre-condition: User has uploaded the code.  Post-condition: System will be able to determine whether the code is efficient.  Success Scenario:  4.1 System is able to analyse the code.  4.2 System is able to compare between the memory allocated and the memory used.  4.3 System is able to successfully deliver the result.  Exceptional Scenario:  4.3.a System fails to deliver the result if there is no internet connection. |

|  |
| --- |
| **5.** **Use case: Accepts the code uploaded by the user** |
| Actor: Server  Pre-condition: User has uploaded the code.  Post-condition: Server will process the code.  Success Scenario:  5.1 Server is able to accept the c-program that will be uploaded.  Exceptional Scenario:  5.1.a Server gives error message if the constructs used in the program are not of C but its extension is \*.c. |

|  |
| --- |
| **6.** **Use case: Processes the code using our program** |
| Actor: Server  Pre-condition: Server has accepted the code.  Post-condition: Server will be able to process the code.  Success Scenario:  6.1 System is able to process the code.  Exceptional Scenario:  6.1.a System may fail to process the code if there is an interruption to the process of analysis. |

|  |
| --- |
| **7.** **Use case: Displays the result.** |
| Actor: Server  Pre-condition: Server has analysed the code.  Post-condition: Server will be able to successfully display the result.  Success Scenario:  7.1 System is able to analyse the code.  Exceptional Scenario:  7.1.a System may fail to give the result if there is no internet connection for connecting to cloud. |

**3.3.3 Non-functional Requirements:**

1. **Portability:** The software is user-friendly and can be used by any user in any operating system but must have Eclipse installed in it (for Plug-in).
2. **Usability:** The software is easy to use as it takes a c program as an input and the analysis is based on the coding standards and optimization techniques of C language (analysis will be done by the System).
3. **Response Time:** The software gives quick result to the user in 5-10 milliseconds to be precise depends on the length of the program as well).
4. **Graphical User Interfaces**: The GUI is very simple and user friendly as it is just an interface between user and the system.

**3.4 Software requirement specifications:**

3.4.1 Software requirement specifications

3.4.1.1 Antlr 4

3.4.1.2 Eclipse Neon IDE

3.4.1.3 AWS Cloud

**3.5 Design Constraints/Assumptions/Dependencies:**

**3.5.1 Design Constraints:**

* Input should be a C program.
* Response time of tool depends on the speed of Internet.

**3.5.2 Assumptions:**

* User is well versed with C programming concepts and implementation.
* User is thorough with the Evaluation scheme.
* Internet is connected before using the tool.
* User is well versed with Java as programming language.

**3.5.3 Dependencies:**

* Wi-Fi (Internet).
* Java inbuilt libraries.
* Java programming concepts.
* C Program as input.

**3.6 Acceptance test plan**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case no. | Input | Expected Output | Actual Output |
| 1. | int a,b,c,d,e,xyz;  File \*fp1;  int func(int a, int b){  int r,s;  e=1;  for(i=0;i<func(a,b);i++){  printf("hey");  r=9;  }  }  func1(int a, int b){  int s;  e=1;  a\*=124;  return a+b;  }  main()  {  int \*h,\*f,\*y;  File \*fp;  float hi;  int x,n,e,p,q;  char l[10];  hi = (float)n;  n = (int)hi;  h = (int\*) malloc(100 \* sizeof(int));  f = (float\*) calloc(25, sizeof(float));  y = (int\*) realloc(10\*sizeof(rec));  fp=fopen("hey.txt");  fp1=fopen("hey.txt");  free(h);  fclose(fp);  a=b+3;  b=c+3;  a=a+1;  a=a+3;  a\*=124;  a\*=128;  a\*=b;  a/=124;  a/=1;  a/=b;  a=128\*d;  a=(a+b)\*128;  b=c\*e;  a=b/124;  a=b/128;  b=c/e;  if(a){  b=1;  }else{  c=1;  }    if(a){  b=1;  }else{  c=1;  d=1;  }    if(a==1){  b=1;  }else if(d==1){  c=1;  }    if(a==1){  b=1;  x=0;  }else if(d==1){  c=1;  }    if(a==1){  a=1;  }else if(a==2){  b=1;  }else if(a==3){  c=1;  }else if(a==4){  d=1;  }    if(a==1){  a=1;  }else if(b==2){  b=1;  }else if(n==3){  c=1;  }else if(e==4){  d=1;  }    switch(a){  case 1: printf("hi");  break;  case 2: printf("hello");  break;  default:break;  }    int i;    for(i=0;i<func(a,b);i++){  printf("hey");  b=a;  a=7;  }  for(i=0;i<strlen(l);i++){  printf("hey");    }  for(i=0;i<(a+b);i++){  printf("hey");  a=5;  }  for(i=0;i<a;i++)  printf("hey");    for(i=0;i<func(a,b);i++){  printf("hey");  a-=9;  }  for(i=0;i<strlen(l);i++){  printf("hey");  l="hi";  }  for(i=0;i<(a+b);i++){  printf("hey");  }  for(i=0;i<a;i++)  printf("hey");    while(i<func(a,b)){  printf("hey");  b=a;  a=7;  }    while(i<strlen(l)){  printf("hey");    }  while(a){  a=9;  printf("hey");  }    do{  c=0;  }while(i>a+b);    do{  i=1;    }while(i<func(a));    do{    a=l;  }while(i<strlen(l));    do{  a=1;  b=0;  }while(i>a+b);  } | assignIneff 3.0  loopsIneff 8.0  scopeOfVarIneff 0.0  varDeclNoyUsedIneff 4.0  ifToTernIneff 1.5  ifToSwitchIneff 1.5  mulDivIneff 6.0  pointerIneff 4.0  typecastIneff 1.0  loop is inefficuent Line: 6 multilpication or division inefficuent Line: 40 Multiplication or division is inefficient Line: 45 Multiplication or division is inefficient Line: 46 Multiplication or division is inefficient Line: 49 if to ternary inefficuent Line: 51 if can be replaced with switch Line: 77 loop is inefficuent Line: 112 loop is inefficuent Line: 143 loop is inefficuent Line: 161 Assignment is inefficuent Line: 37 Assignment is inefficuent Line: 38 type cast inefficuent Line: 1 | assignIneff 3.0  loopsIneff 8.0  scopeOfVarIneff 0.0  varDeclNoyUsedIneff 4.0  ifToTernIneff 1.5  ifToSwitchIneff 1.5  mulDivIneff 6.0  pointerIneff 4.0  typecastIneff 1.0  loop is inefficuent Line: 6 multilpication or division inefficuent Line: 40 Multiplication or division is inefficient Line: 45 Multiplication or division is inefficient Line: 46 Multiplication or division is inefficient Line: 49 if to ternary inefficuent Line: 51 if can be replaced with switch Line: 77 loop is inefficuent Line: 112 loop is inefficuent Line: 143 loop is inefficuent Line: 161 Assignment is inefficuent Line: 37 Assignment is inefficuent Line: 38 type cast inefficuent Line: |

|  |  |  |  |
| --- | --- | --- | --- |
| 2. | /\*Author: Goutam Naik (1239)  Date: 10-3-17  Program to check whether a number is palindrome or not\*/  #include<stdio.h>  #include<math.h>  void palindrome(int n, int d) //function  {  int r,m=n,i=(d-1),s=0;    while(i>=0)  {r=n%10;  s=s+(r\*(pow(10,i)));  n=n/10;  i--;  }  if(s==m)  {printf("%d is a palindrome\n",m);}  else  {printf("%d is not a palindrome\n ",m);}  }  main()  { int num,dig,count=0;  printf("Enter the number to be checked\n");  scanf("%d",&num); //input the number  int number=num;    while(num!=0)  {int rem=num%10;  num=num/10;  count+=1;  } //loop to claculate no. of digits  dig=count;  palindrome(number,dig); //function call  } | assignIneff 0.0  loopsIneff 0.0  scopeOfVarIneff 0.0  varDeclNoyUsedIneff 1.0  ifToTernIneff 1.5  ifToSwitchIneff 0.0  mulDivIneff 0.0  pointerIneff 0.0  typecastIneff 0.0  if to ternary inefficient Line: 18 | assignIneff 0.0  loopsIneff 0.0  scopeOfVarIneff 0.0  varDeclNoyUsedIneff 1.0  ifToTernIneff 1.5  ifToSwitchIneff 0.0  mulDivIneff 0.0  pointerIneff 0.0  typecastIneff 0.0  if to ternary inefficient Line: 18 |

|  |  |  |  |
| --- | --- | --- | --- |
| 3. | //pass2.c  #include<stdio.h>  #include<conio.h>  #include<string.h>  #include<ctype.h>  main()  {  FILE \*fint,\*ftab,\*flen,\*fsym;  int op1[10],txtlen,txtlen1,i,j=0,len;  char add[5],symadd[5],op[5],start[10],  temp[30],line[20],  label[20],mne[10],operand[10],  symtab[10],  opmne[10];  clrscr();  fint=fopen("input.txt","r");  flen=fopen("length.txt","r");  ftab=fopen("optab.txt","r");  fsym=fopen("symbol.txt","r");  fscanf(fint,"%s%s%s%s"  ,add,label,mne,operand);  if(strcmp(mne,"START")==0)  {  strcpy(start,operand);  fscanf(flen,"%d",&len);  }  printf("H^%s^%s^%d\nT^00%s^"  ,label,start,len,start);  fscanf(fint,"%s%s%s%s"  ,add,label,mne,operand);  while(strcmp(mne,"END")!=0)  {  fscanf(ftab,"%s%s",opmne,op);  while(!feof(ftab))  {  if(strcmp(mne,opmne)==0)  {  fclose(ftab);  fscanf(fsym,"%s%s",symadd,symtab);  while(!feof(fsym))  {  if(strcmp(operand,symtab)==0)  {  printf("%s%s^",op,symadd);  break;  }  else  fscanf(fsym,"%s%s",symadd,symtab);  }  break;  }  else  fscanf(ftab,"%s%s",opmne,op);  }  if((strcmp(mne,"BYTE")==0)||  (strcmp(mne,"WORD")==0))  {  if(strcmp(mne,"WORD")==0)  printf("0000%s^",operand);  else  {  len=strlen(operand);  for(i=2;i<strlen(operand);i++)  {  printf("%d",operand[i]);  printf(i);  }  printf("^");  }  }  fscanf(fint,"%s%s%s%s"  ,add,label,mne,operand);  ftab=fopen("optab.txt","r");  fseek(ftab,SEEK\_SET,0);  }  printf("\nE^00%s",start);  fclose(fint);  fclose(ftab);  fclose(fsym);  fclose(flen);  fclose(fout);  getch();  } | assignIneff 0.0  loopsIneff 0.0  scopeOfVarIneff 0.0  varDeclNoyUsedIneff 7.0  ifToTernIneff 0.0  ifToSwitchIneff 0.0  mulDivIneff 0.0  pointerIneff 0.0  typecastIneff 0.0 | assignIneff 0.0  loopsIneff 0.0  scopeOfVarIneff 0.0  varDeclNoyUsedIneff 7.0  ifToTernIneff 0.0  ifToSwitchIneff 0.0  mulDivIneff 0.0  pointerIneff 0.0  typecastIneff 0.0 |

**4. System Design**

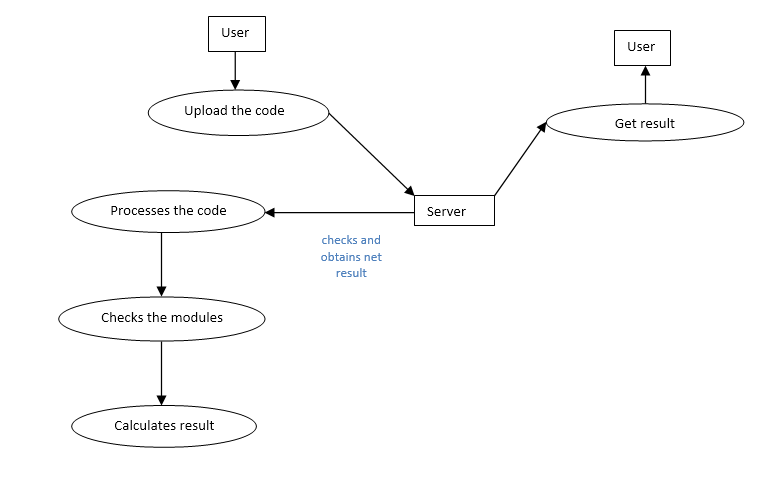
* 1. **Data Structures used and Justification**:

The data structures used in our java programs are:-

* **Lists**: - This data structure is used to store all the variables used in the program in accordance with parameter being checked.

Since the entire analysis is based upon a pattern of usage, sometimes it’s necessary that until we parse the entire program we cannot match up to a pattern, hence most of the times list is used to store all the variables/patterns used which is further used for analysis.

**4.2 Level 0 DFD**



**Figure 3**

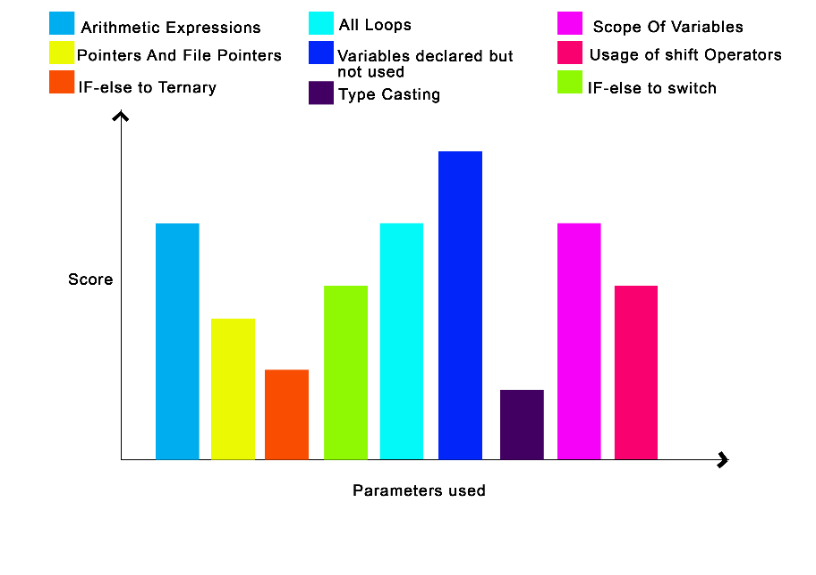
Our Software tool depends on the input i.e, C-program and on the internet to get connected to cloud wherein they will upload this C-program and get results in the form of bar graph and inefficiency percentage.

Form figure 3, User will first get connected to the cloud by using the URL provided. They will select the C-program to be uploaded to the server among the list of programs of which the result is to be found out. The server will process the code and check for modules which are implemented in inefficient ways. The modules which are checked are Assignment expressions, optimised way of implementing the for, while loops, implementing switch statements except the if-else ladder which takes less memory, usage of ternary operators where only one if and else statement exists, whether the allocated memory is freed in case of pointers and file pointers or no, Type casting etc.

Using all the modules the result will be calculated in terms of inefficiency percentage and this result will be given to the server. Using this inefficiency percentage, the server generates a bar graph which describes what are the inefficient ways are being implemented in the code, so that by considering these inefficient ways the users can further improve their coding skills and standards to have an efficient program.

**4.2 GUI Wireframe**





**Figure 4**

**Figure 4**

Wireframes are static mock-ups used to present GUI-Elements for different page types or page functions.

From figure 4, The GUI wireframe of out tool contains a browse button to select the files to be uploaded and once the file is selected and then on clicking the button submit the processing of code starts and then the results in inefficiency percentage is displayed in the text box and the bar graph will be displayed as per the parameters used and pertaining score for those parameters will be displayed in graph a well as in the list beside the graph so that user can easily understand the marks scored.

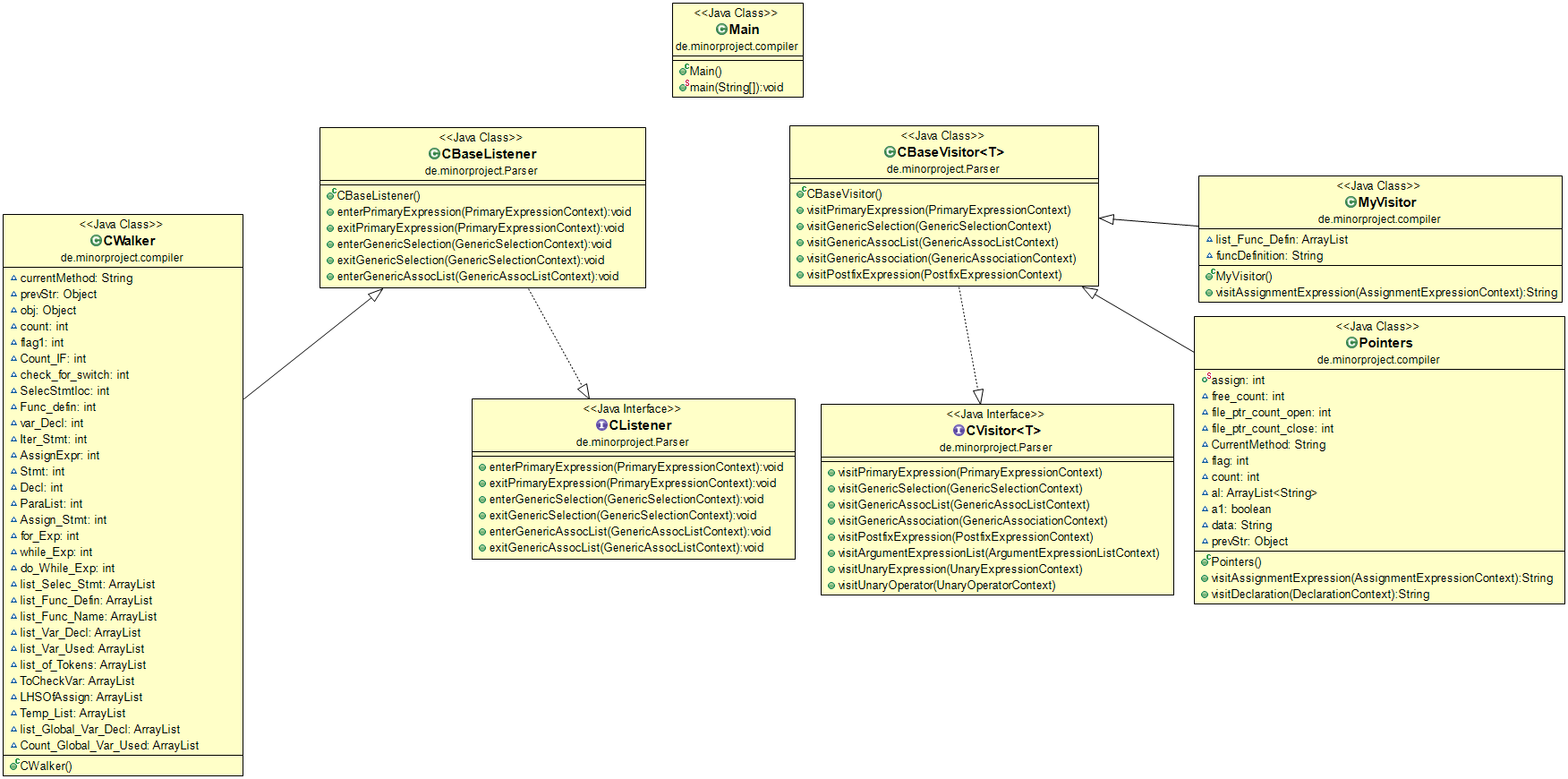
**4.3 Class Diagram**

Class diagram specify the static structure of the system

Classes are the basic building blocks of an OO system as classes are the implementation units also.

Class diagram is the central piece in an OO design. It specifies.

1. Classes in the system
2. Association between classes
3. Subtype, super-type and relationship



**Figure 5**

From figure 5,

|  |  |
| --- | --- |
| Function | Description |
| CBaseVisitor() | Visits the leaf nodes of the tree where the identifiers are variables are found. |
| MyVisitor() | Analysis of C-program efficiency using a Java program is done. |
| Pointers() | Used for the analysis of pointers and files pointers. |
| CBaseListener() | defines a listener that would take a C-program as input |
| CWalker() | Parses the tree generated using the C-Grammar. |
| VisitUnaryExpression(UnaryExpressionContext) | Visits only unary expression subtree among the whole tree. |
| VisitUnaryOperator(UnaryOperatorContext) | Visits only unary operator subtree among the whole tree. |
| enterPrimaryExpression(PrimaryExpressionContext) | Enters only primary expression subtree among the whole tree nodes. |
| exitPrimaryExpression(PrimaryExpressionContext) | Exists from primary expression subtree among the whole tree nodes after visiting all nodes pertaining to primary expression. |

**4.4** **Class coupling matrix and explanation**

Class coupling matrix specifies the relationship between the classes.

The relationship between the classes may be,

Inheritance: If a class has inherited services of other classes

Association: If a class needs services from any other classes.

Aggregation: If a class is composed of many classes.

Generalization/Specialization: If there is a superclass and subclass relationship between

the classes.

In our project, we have identified the association relationships among the classes which is shown as below.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **C Walker** | **CBaseVisitor** | **CBaseListener** | **MyVisitor** | **Pointers** | **CListener** | **CVisitor** |
| **C Walker** | - | - | Extends | - | - | - | - |
| **CBaseVisitor** | - | - | - | - | - | - | Implements |
| **CBaseListener** | - | - | - | - | - | Implements | - |
| **MyVisitor** | - | Extends | - | - | - | - | - |
| **Pointers** | - | Extends | - | - | - | - | - |
| **CListener** | - | - | - | - | - | - | - |
| **CVisitor** | - | - | - | - | - | - | - |

1. **Implementation**

**5.1 Implementation**

On choosing the domain systems, we took long to choose a problem statement, after being driven by many problem statements, we finally chose a DRDO problem statement “Programmer’s Productivity Analyser”. It took us long to understand how ANTLR works, how it generates a tree, and finally how can we visit that tree in our code, which are inherited from ANTLR generated java classes. After figuring out how to use listener’s and visitor’s, we were able to walk through the tree, get into a sub tree, like iteration statement, selection statement etc. get to know the pattern a user used to code, and if it matches to an inefficient pattern, the user is notified with inefficiency flag along with line number in the code snippet, giving him a chance to rectify, so the main moto of developing the tool is to help a programmer to find his week points in his coding so that he can develop good programming skills, and also an organisation or a company can test their employees programs and decide on what a programmer’s productivity is, not all the organisations/companies need the same parameter as we used upon, but the ANTLR technology is flexible enough to accommodate the changes, since the tree generated remains the same, the class inheriting the generated classes can be designed as per the needs of the company.

The next task was about deployment, any software tool cannot be delivered as a mere code to the user, hence we chose to deploy our tool on cloud, so that any user can upload his/her code from the place he/she is in and get the results, We hence started with learning how to use servlets to deploy our code. After a long way along learning we were able to run our class of code from a servlet code, through which we were able to give the user a user friendly environment where in he gets the results in the form of a graph.

The idea of this entire process focuses majorly on how we can help a programmer to know his week points to develop a good programming skills, the parameters we’ve considered is being detecting by the compiler (function calls in condition expression without changing parameters) and is also fixed at run time, since this task is done in the background, the programmer is not aware of it. He continues to write the same shabby code. Hence our focus in entirely on the programmer and letting him know where he can go wrong, and of course where he is supposed to correct.

**5.2 Introduction**

**Pair programming:**

1. Code is written by pair of programmers rather than individuals.
2. The pair together design data structures, strategies, etc.
3. One person types the code, the other actively reviews what is being typed
4. Errors are pointed out and together solutions are formulated
5. Roles are reversed periodically

**5.3 Brief description of modules:**

**5.2.1** **CWalker**- Parses the tree generated using the C-Grammar.

**5.2.2 CBaseListener**- defines a listener that would take a C-program as input.

**5.2.3 CBaseVisitor –**Visits the leaf nodes of the tree where the identifiers or variables are found.

**5.2.4 CListener -**Provides information about the c-program that is parsed.

**5.4.5 CVisitor–** Provides information about the tree nodes visited of a c-program that is parsed.

**5.3.6 MyVisitor –**Analysis of C-program efficiency using a Java program is done.

**5.3.7 Pointers –** Used for the analysis of pointers and files pointers

**6. Testing**

**6.1 Introduction**

Software testing is the process of analysing a software item to detect the differences between existing and required conditions (that is, bugs) and to evaluate the features of the software item.

White box testing was carried out since the testing was carried out by the programmer itself, all the test cases were written based on the execution of statements in the program, to see whether all of the statements will execute at least once in different scenarios, since the tester and33 programmer were the same, whenever there was an error which was not handled, it was fixed at the same time, since all the internal functionality was known to the tester.

Unit Testing:

Unit testing is the testing of individual hardware or software units or groups of related units. Unit test begins at the vertex of the spiral and concentrates on each unit of the software as implemented in the source code.

It focuses on,

1. Internal processing logic
2. Data structures within the boundaries of a component.

The design of unit tests can be performed before coding begins or after source code has been generated. A review of design information provides guidance for establishing test cases that are likely to uncover errors in each of the categories.

Each test case should be coupled with a set of expected results.

But our app does not take the input from the users. So unit test is done for separate program.

Alpha – Beta testing:

Alpha Test

1. Conducted at the developer’s site by end users.
2. Tests are conducted in a controlled environment.

Beta Test

1. Conducted by the customer at their sites.
2. Conducted in a real environment that cannot be controlled by the developer.
3. Customer sends the report of errors & that can be corrected by developer and then the product is released.

**6.2 Module test plan and Test Cases**

**6.2.1. Module test plan for Programmer’s Efficiency Analyser**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Id** | **Input** | **Expected Output** | **Actual Output** |
| **Module 1: If-else to Switch** | | | |
| 1.01 | int main(){  int ch=0;  printf("Enter the number of the day\n");  scanf("%d",&ch);  if(ch==1)  printf("it's sunday\n");  else if(ch==2)  printf("it's monday\n");  else if(ch==3)  printf("it's tuesday\n");  else if(ch==4)  printf("it's wednesday\n");  else if(ch==5)  printf("it's thursday\n");  else if(ch==6)  Printf("it's friday\n");  else if(ch==7)  printf("it's saturday\n");  else  printf("wrong choice\n");  } | assignEff 0  assignNotEff 0  loopsEff 0  loopsNotEff 0  scopeOfVarEff 0  scopeOfVarNotEff 0  varDeclUsedEff 1  varDeclNotUsedEff 2  ifEff 1  ifToSwitchNotEff 0  ifToTernNotEff 0  mulDivExpEff 0  mulDivExpNotEff 0  assignIneff 0.0  loopsIneff 0.0  scopeOfVarIneff 0.0  varDeclNoyUsedIneff 2.0  ifToTernIneff 0.0  ifToSwitchIneff 0.0  mulDivIneff 0.0 | assignEff 0  assignNotEff 0  loopsEff 0  loopsNotEff 0  scopeOfVarEff 0  scopeOfVarNotEff 0  varDeclUsedEff 1  varDeclNotUsedEff 2  ifEff 1  ifToSwitchNotEff 0  ifToTernNotEff 0  mulDivExpEff 0  mulDivExpNotEff 0  assignIneff 0.0  loopsIneff 0.0  scopeOfVarIneff 0.0  varDeclNoyUsedIneff 2.0  ifToTernIneff 0.0  ifToSwitchIneff 0.0  mulDivIneff 0.0    Along with Bar Graph and total inefficiency percentage. |

|  |  |  |  |
| --- | --- | --- | --- |
| 1.02 | int main(){  int ch,a,d,sum;  printf("1.add\n 2.sub\n 3.mul\n 4.div\n");  printf("Enter your choice:")  scanf("%d",&ch);  printf("enter the numbers a and b:")  scanf("%d%d",&a,&b);  if(ch==1)  sum=a+b;  else if(ch==2)  sum=a-b;  else if(ch==3)  sum=a\*b;  else if(ch==4)  sum=a/b;  else  printf("wrong choice\n");  } | assignEff 4  assignNotEff 0  loopsEff 0  loopsNotEff 0  scopeOfVarEff 0  scopeOfVarNotEff 0  varDeclUsedEff 3  varDeclNotUsedEff 1  ifEff 1  ifToSwitchNotEff 0  ifToTernNotEff 0  mulDivExpEff 0  mulDivExpNotEff 0  assignIneff 0.0  loopsIneff 0.0  scopeOfVarIneff 0.0  varDeclNoyUsedIneff 1.0  ifToTernIneff 0.0  ifToSwitchIneff 0.0  mulDivIneff 0.0 | assignEff 4  assignNotEff 0  loopsEff 0  loopsNotEff 0  scopeOfVarEff 0  scopeOfVarNotEff 0  varDeclUsedEff 3  varDeclNotUsedEff 1  ifEff 1  ifToSwitchNotEff 0  ifToTernNotEff 0  mulDivExpEff 0  mulDivExpNotEff 0  assignIneff 0.0  loopsIneff 0.0  scopeOfVarIneff 0.0  varDeclNoyUsedIneff 1.0  ifToTernIneff 0.0  ifToSwitchIneff 0.0  mulDivIneff 0.0 |
| **Module 2: If-else to ternary** | | | |
| 2.01 | //if else to ternaary  int main(){  int num;  printf("Enter the number:");  scanf("%d",&num);  if(num%2==0)  printf("divisible by 2");  else  printf("not divisible by 2");  } | assignEff 0  assignNotEff 0  loopsEff 0  loopsNotEff 0  scopeOfVarEff 0  scopeOfVarNotEff 0  varDeclUsedEff 1  varDeclNotUsedEff 0  ifEff 0  ifToSwitchNotEff 0  ifToTernNotEff 1  mulDivExpEff 0  mulDivExpNotEff 1  assignIneff 0.0  loopsIneff 0.0  scopeOfVarIneff 0.0  varDeclNoyUsedIneff 0.0  ifToTernIneff 0.0  ifToSwitchIneff 1.5  mulDivIneff 1.0 | assignEff 0  assignNotEff 0  loopsEff 0  loopsNotEff 0  scopeOfVarEff 0  scopeOfVarNotEff 0  varDeclUsedEff 1  varDeclNotUsedEff 0  ifEff 0  ifToSwitchNotEff 0  ifToTernNotEff 1  mulDivExpEff 0  mulDivExpNotEff 1  assignIneff 0.0  loopsIneff 0.0  scopeOfVarIneff 0.0  varDeclNoyUsedIneff 0.0  ifToTernIneff 0.0  ifToSwitchIneff 1.5  mulDivIneff 1.0 |

|  |  |  |  |
| --- | --- | --- | --- |
| 2.02 | //if else to ternaary  int main(){  int num,res;  printf("enter the number");  scanf("%d",&num);  if(num%10==0){  printf("the number is divisible by 10");  res=num\*num;  }  else{  printf("the number is not divisible by 10");  res=num\*num\*num;  }  } | assignEff 2  assignNotEff 0  loopsEff 0  loopsNotEff 0  scopeOfVarEff 0  scopeOfVarNotEff 0  varDeclUsedEff 2  varDeclNotUsedEff 0  ifEff 1  ifToSwitchNotEff 0  ifToTernNotEff 0  mulDivExpEff 1  mulDivExpNotEff 0  assignIneff 0.0  loopsIneff 0.0  scopeOfVarIneff 0.0  varDeclNoyUsedIneff 0.0  ifToTernIneff 0.0  ifToSwitchIneff 0.0  mulDivIneff 0.0 | assignEff 2  assignNotEff 0  loopsEff 0  loopsNotEff 0  scopeOfVarEff 0  scopeOfVarNotEff 0  varDeclUsedEff 2  varDeclNotUsedEff 0  ifEff 1  ifToSwitchNotEff 0  ifToTernNotEff 0  mulDivExpEff 1  mulDivExpNotEff 0  assignIneff 0.0  loopsIneff 0.0  scopeOfVarIneff 0.0  varDeclNoyUsedIneff 0.0  ifToTernIneff 0.0  ifToSwitchIneff 0.0  mulDivIneff 0.0 |
| **Module 3: Loops** | | | |
| 3.01 | // for loops  int main(){  int search,size,arr[10];  printf("enter the size of the array\n");  scanf("%d",&size);  printf("enter the elements of array:\n");  for(i=0;i<size;i++){  scanf("%d",arr[i]);  }printf("enter the element to be searched:\n");  scanf("%d",&search);  for(i=0;i<size;i++){  if(arr[i]==search){  found=1;  break;  }  }if(found==1)  printf("key found at pos: ",i+1);  else  printf("key not found");  } | assignEff 3  assignNotEff 0  loopsEff 2  loopsNotEff 0  scopeOfVarEff 0  scopeOfVarNotEff 0  varDeclUsedEff 5  varDeclNotUsedEff 1  ifEff 0  ifToSwitchNotEff 0  ifToTernNotEff 1  mulDivExpEff 0  mulDivExpNotEff 0  assignIneff 0.0  loopsIneff 0.0  scopeOfVarIneff 0.0  varDeclNoyUsedIneff 1.0  ifToTernIneff 0.0  ifToSwitchIneff 1.5  mulDivIneff 0.0 | assignEff 3  assignNotEff 0  loopsEff 2  loopsNotEff 0  scopeOfVarEff 0  scopeOfVarNotEff 0  varDeclUsedEff 5  varDeclNotUsedEff 1  ifEff 0  ifToSwitchNotEff 0  ifToTernNotEff 1  mulDivExpEff 0  mulDivExpNotEff 0  assignIneff 0.0  loopsIneff 0.0  scopeOfVarIneff 0.0  varDeclNoyUsedIneff 1.0  ifToTernIneff 0.0  ifToSwitchIneff 1.5  mulDivIneff 0.0 |

|  |  |  |  |
| --- | --- | --- | --- |
| 3.02 | //for loops  int main(){  char l[10],copy[10];  int i=0;  printf("enter the string:\n");  scanf("%s",l);  for(i=0;i<strlen(l);i++){  copy[i]=l[i];  }  } | assignEff 2  assignNotEff 0  loopsEff 0  loopsNotEff 1  scopeOfVarEff 0  scopeOfVarNotEff 0  varDeclUsedEff 9  varDeclNotUsedEff 2  ifEff 0  ifToSwitchNotEff 0  ifToTernNotEff 0  mulDivExpEff 0  mulDivExpNotEff 0  assignIneff 0.0  loopsIneff 2.0  scopeOfVarIneff 0.0  varDeclNoyUsedIneff 2.0  ifToTernIneff 0.0  ifToSwitchIneff 0.0  mulDivIneff 0.0 | assignEff 2  assignNotEff 0  loopsEff 0  loopsNotEff 1  scopeOfVarEff 0  scopeOfVarNotEff 0  varDeclUsedEff 9  varDeclNotUsedEff 2  ifEff 0  ifToSwitchNotEff 0  ifToTernNotEff 0  mulDivExpEff 0  mulDivExpNotEff 0  assignIneff 0.0  loopsIneff 2.0  scopeOfVarIneff 0.0  varDeclNoyUsedIneff 2.0  ifToTernIneff 0.0  ifToSwitchIneff 0.0  mulDivIneff 0.0 |
| 3.03 | //while loops  main(){  char l[10],copy[10];  int i;  printf("enter the string:\n");  scanf("%s",l);  i=0;  while(i<strlen(l)){  copy[i]=l[i];  i++;  }  } | assignEff 2  assignNotEff 0  loopsEff 0  loopsNotEff 1  scopeOfVarEff 0  scopeOfVarNotEff 0  varDeclUsedEff 7  varDeclNotUsedEff 2  ifEff 0  ifToSwitchNotEff 0  ifToTernNotEff 0  mulDivExpEff 0  mulDivExpNotEff 0  assignIneff 0.0  loopsIneff 2.0  scopeOfVarIneff 0.0  varDeclNoyUsedIneff 2.0  ifToTernIneff 0.0  ifToSwitchIneff 0.0  mulDivIneff 0.0 | assignEff 2  assignNotEff 0  loopsEff 0  loopsNotEff 1  scopeOfVarEff 0  scopeOfVarNotEff 0  varDeclUsedEff 7  varDeclNotUsedEff 2  ifEff 0  ifToSwitchNotEff 0  ifToTernNotEff 0  mulDivExpEff 0  mulDivExpNotEff 0  assignIneff 0.0  loopsIneff 2.0  scopeOfVarIneff 0.0  varDeclNoyUsedIneff 2.0  ifToTernIneff 0.0  ifToSwitchIneff 0.0  mulDivIneff 0.0 |

|  |  |  |  |
| --- | --- | --- | --- |
| 3.04 | //while loops  main(){  char l[10],copy[10];  int i;  printf("enter the string:\n");  scanf("%s",l);  i=0;  do{  copy[i]=l[i];  i++;  }while(i<strlen(l));  } |  |  |
| **Module 4: Scope Of Variables** | | | |
| 4.01 | //scope of var  int a,b,c,d,e;  int func(int a, int b){  int r,s;  e=1;  for(i=0;i<func(a,b);i++){  printf("hey");  r=9;  }  }  func1(int a, int b){  int s;  e=1;  a\*=124;  return a+b;  }  main()  {  int x,n,e,p,q;  char l;  a=b+3;  b=c+3;  a=a+1;  } | assignEff 7  assignNotEff 1  loopsEff 0  loopsNotEff 1  scopeOfVarEff 1  scopeOfVarNotEff 4  varDeclUsedEff 5  varDeclNotUsedEff 8  ifEff 0  ifToSwitchNotEff 0  ifToTernNotEff 0  mulDivExpEff 1  mulDivExpNotEff 0  assignIneff 1.0  loopsIneff 2.0  scopeOfVarIneff 6.0  varDeclNoyUsedIneff 8.0  ifToTernIneff 0.0  ifToSwitchIneff 0.0  mulDivIneff 0.0 | assignEff 7  assignNotEff 1  loopsEff 0  loopsNotEff 1  scopeOfVarEff 1  scopeOfVarNotEff 4  varDeclUsedEff 5  varDeclNotUsedEff 8  ifEff 0  ifToSwitchNotEff 0  ifToTernNotEff 0  mulDivExpEff 1  mulDivExpNotEff 0  assignIneff 1.0  loopsIneff 2.0  scopeOfVarIneff 6.0  varDeclNoyUsedIneff 8.0  ifToTernIneff 0.0  ifToSwitchIneff 0.0  mulDivIneff 0.0 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Module 5: Assignment Expressions** | | | |
| 5.01 | //assign  int a,b,c,d,e;  func1(int a, int b){  int s;  e=1;  a\*=124;  return a+b;  }  main()  {  int x,n,e,p,q;  char l;  a=b+3;  b=c+3;  a=a+1;  a\*=124;  a\*=128;  a\*=b;  a/=124;  a/=1;  a/=b;  a=128\*d;  a=(a+b)\*128;  b=c\*e;  a=b/124;  a=b/128;  b=c/e;  } | assignEff 16  assignNotEff 1  loopsEff 0  loopsNotEff 0  scopeOfVarEff 0  scopeOfVarNotEff 5  varDeclUsedEff 3  varDeclNotUsedEff 6  ifEff 0  ifToSwitchNotEff 0  ifToTernNotEff 0  mulDivExpEff 5  mulDivExpNotEff 4  assignIneff 1.0  loopsIneff 0.0  scopeOfVarIneff 7.5  varDeclNoyUsedIneff 6.0  ifToTernIneff 0.0  ifToSwitchIneff 0.0  mulDivIneff 4.0 | assignEff 16  assignNotEff 1  loopsEff 0  loopsNotEff 0  scopeOfVarEff 0  scopeOfVarNotEff 5  varDeclUsedEff 3  varDeclNotUsedEff 6  ifEff 0  ifToSwitchNotEff 0  ifToTernNotEff 0  mulDivExpEff 5  mulDivExpNotEff 4  assignIneff 1.0  loopsIneff 0.0  scopeOfVarIneff 7.5  varDeclNoyUsedIneff 6.0  ifToTernIneff 0.0  ifToSwitchIneff 0.0  mulDivIneff 4.0 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Module 6: Variables declared but not used** | | | |
| 6.01 | //var declared not used  //mul and div  main()  {  int a,b,c,x,n,e,p,q;  char l;  a=b+3;  b=c+3;  a=a+1;  a\*=124;  a\*=128;  a\*=b;  a/=124;  a/=1;  a/=b;  a=128\*d;  a=(a+b)\*128;  b=c\*e;  a=b/124;  a=b/128;  b=c/e;  } | assignEff 14  assignNotEff 1  loopsEff 0  loopsNotEff 0  scopeOfVarEff 0  scopeOfVarNotEff 0  varDeclUsedEff 4  varDeclNotUsedEff 5  ifEff 0  ifToSwitchNotEff 0  ifToTernNotEff 0  mulDivExpEff 4  mulDivExpNotEff 4  assignIneff 1.0  loopsIneff 0.0  scopeOfVarIneff 0.0  varDeclNoyUsedIneff 5.0  ifToTernIneff 0.0  ifToSwitchIneff 0.0  mulDivIneff 4.0 | assignEff 14  assignNotEff 1  loopsEff 0  loopsNotEff 0  scopeOfVarEff 0  scopeOfVarNotEff 0  varDeclUsedEff 4  varDeclNotUsedEff 5  ifEff 0  ifToSwitchNotEff 0  ifToTernNotEff 0  mulDivExpEff 4  mulDivExpNotEff 4  assignIneff 1.0  loopsIneff 0.0  scopeOfVarIneff 0.0  varDeclNoyUsedIneff 5.0  ifToTernIneff 0.0  ifToSwitchIneff 0.0  mulDivIneff 4.0 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Module 7:Pointers And File Pointers** | | | |
| 7.01 | //File pointers and pointers  struct rec {  char \*date;  float \*yld;  };  void func(int \*a,int \*b)  {  int \*c;  }  main ()  {  struct rec \*r;  int \*p, \*a,\*b,\*c,d,e;  FILE \*fp;  fp=fopen("1.txt","w+");  p = (int\*) malloc (100 \* sizeof(int));  a = (float\*) calloc (25, sizeof(float));  r = (int\*) realloc (10\*sizeof(rec));  \*c=\*p;  free(p);  free(a);  free(r);  fclose(fp);  fclose(fp1);  } | PointersIneff 0  PointersEff 1  filePointersEff 1  filePointersIneff 1  assignEff 5  assignNotEff 0  loopsEff 0  loopsNotEff 0  scopeOfVarEff 2  scopeOfVarNotEff 6  varDeclUsedEff 31  varDeclNotUsedEff 12  ifEff 0  ifToSwitchNotEff 0  ifToTernNotEff 0  mulDivExpEff 2  mulDivExpNotEff 0 | PointersIneff 0  PointersEff 1  filePointersEff 1  filePointersIneff 1  assignEff 5  assignNotEff 0  loopsEff 0  loopsNotEff 0  scopeOfVarEff 2  scopeOfVarNotEff 6  varDeclUsedEff 31  varDeclNotUsedEff 12  ifEff 0  ifToSwitchNotEff 0  ifToTernNotEff 0  mulDivExpEff 2  mulDivExpNotEff 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Module 8:Type Casting** | | | |
| 7.01 | //Type Casting  main()  {  int a;  float b;  a=10;  b=10.5;  a=(int)b;  } | assignEff 5  assignNotEff 0  loopsEff 0  loopsNotEff 0  scopeOfVarEff 0  scopeOfVarNotEff 0  varDeclUsedEff 2  varDeclNotUsedEff 0  ifEff 0  ifToSwitchNotEff 0  ifToTernNotEff 0  mulDivExpEff 0  mulDivExpNotEff 0  typecasteff 0  typecasNoteff 1 | assignEff 5  assignNotEff 0  loopsEff 0  loopsNotEff 0  scopeOfVarEff 0  scopeOfVarNotEff 0  varDeclUsedEff 2  varDeclNotUsedEff 0  ifEff 0  ifToSwitchNotEff 0  ifToTernNotEff 0  mulDivExpEff 0  mulDivExpNotEff 0  typecasteff 0  typecasNoteff 1 |

**6.3 JUNIT testing**

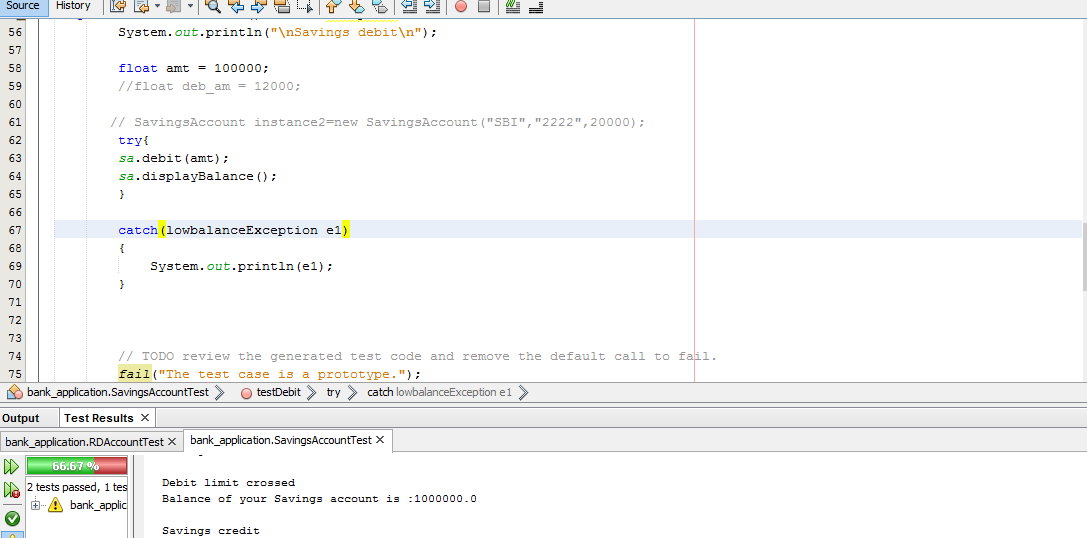
Since in our project we have not taken any input from the user, we have not done JUNIT testing for our project. But we have done the JUNIT testing for the sample code (BANK APPLICATION).

Modules tested in BANK APPLICATION using JUNIT- Credit and debit

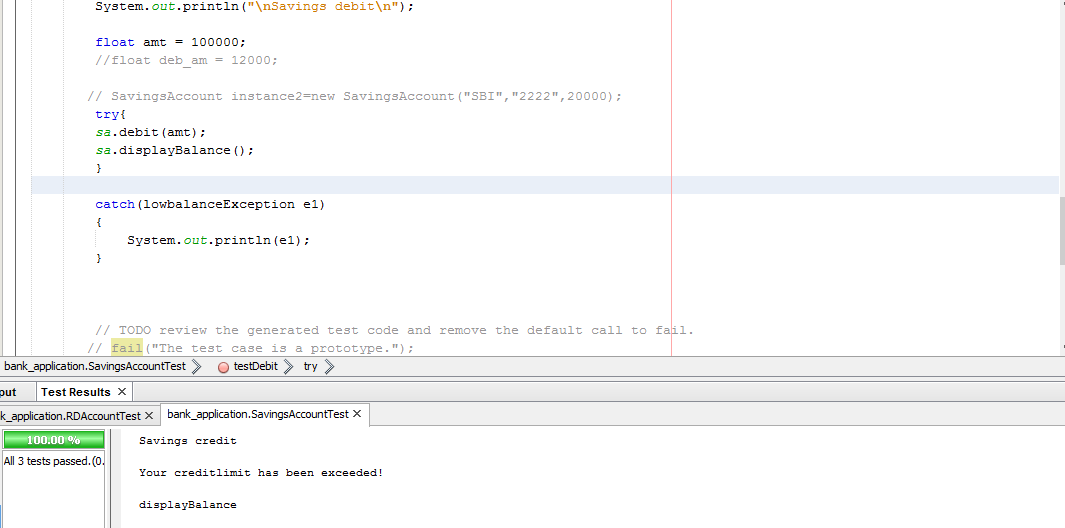
In credit and debit methods, User is given with some specifications. That is, a user cannot credit beyond the credit limit. If this is the case that needs to be handled and if that is tested with JUNIT, then the test case will pass.

OUTPUT is as shown below:

1) Initially test script for method was passed 66.7% because the exception was not handled. (Exception – credit limit and debit limit).



2) After handling exception all the tests were passed.



**6.4 Usability testing report**

Feedback obtained from users of this product derives the below conclusion:

* The software tool is easy to use.
* Knowledge of C coding standards is preferred when the tool is in use.
* Internet speed should be good for quick response.
* The software tool will definitely help people who want to know their inefficient ways of coding and using which they can write better optimized programs in future.
* It helps in situations where better coder is to be picked among all the coders for the program with different approaches for solving it.

**7. Results**

When a user uploads his c program, he’ll get a graph giving him the number of inefficient constructs he has used multiplied by it’s weightage pre-assigned.

He is also given with the line numbers where the construct usage is wrong, so that he’ll be able to fix it. With this a user can get to know his productivity is terms of using constructs efficiently.

**8. Summary of learning:**

We got to learn various C and Java concepts and library that we implemented in the project. We were able to learn more about Cloud and how one programming language can be used to analyse other programming language and also the concepts that we had learnt in “Principles of Compiler Design” like parser, lexer helped us in better understanding of what is to be done. The tree that was generated for C-program as an input using the C-Grammar in “Antlr”, provided great knowledge and learning and better implementation of software tool was possible.

**9. Conclusion and future scope**

Conclusion:

Thus, we have implemented all the features proposed in the beginning in our project successfully with some additional features.

Future scope:

As this project acts as a Programmer’s efficiency analyser to the user, interfacing this software with a “Plagiarism Tester” would make it more helpful to the companies and organizations where they want to make sure that their codes are almost original rather than having functions that have been already used in other programs or from existing programs if the functions have been taken then that can be caught using plagiarism testing. Further the scope can be widened by making the software tool to analyse object oriented programming languages like C++, Java, C# etc.

**10. References/Bibliography**

[1]. <http://www.antlr.org/>

[2]. https://github.com/antlr/antlr4/blob/master/doc/index.md

[2]. <http://www.csd.uoc.gr/~zakkak/hy255/reading/c_coding_style.pdf>

[3]. <http://www.cs.cornell.edu/courses/cs314/2004fa/tutorials/cstyle.pdf>

[4]. http://everywarecloud.eurotech.com/doc/ECDevGuide/latest/3.03-EDC-Client-Java.asp

[5]. <https://www.genuitec.com/connecting-delivery-sharing/>

[6]. <https://www.codeproject.com/tips/893547/how-to-create-your-own-eclipse-ide-plug-in>

**11. Appendix**

A. Glossary

1. Listener – Virtually Interactive Kinetic Intelligence
2. Visitor– Text to Speech

**B. Explanation on JUNIT tool**

JUnit is an open source framework that has been designed for the purpose of writing and running tests in the Java programming language. JUnit was originally written by Erich Gamma and Kent Beck. There are many ways to write test cases. A test case is a code fragment that checks that another code unit (method) works as expected. So if we want an accurate and efficient testing process then using a good testing framework is recommended. JUnit has established a good reputation in this scenario.

JUnit is a regression-testing framework that developers can use to write unit tests as they develop systems. Unit testing belongs to test a single unit of code, which can be a single class for Java. This framework creates a relationship between development and testing. You start coding according to the specification and need and use the JUnit test runners to verify how much it deviates from the intended goal. Typically, in a unit testing, we start testing after completing a module but **JUnit helps us to code and test both during the development**.

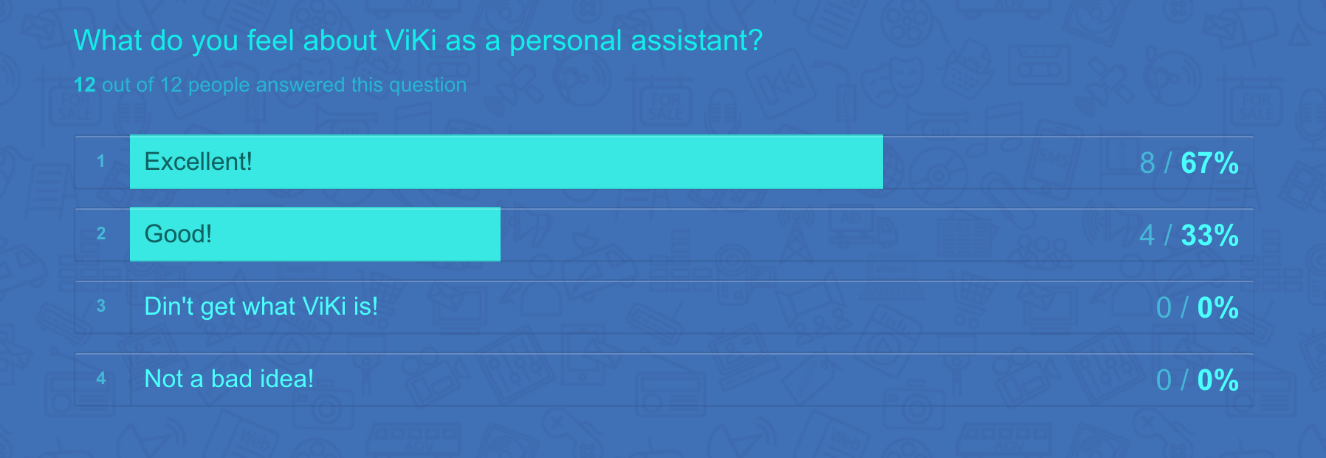
Using a framework, like JUnit, to develop your test cases has a number of advantages, most important being that others will be

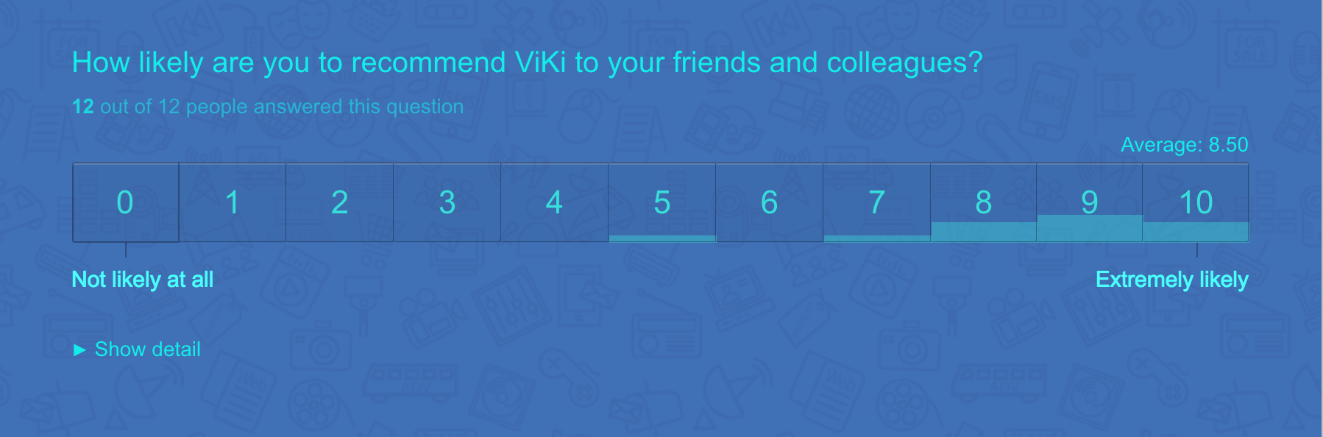
* able to understand test cases and
* Easily write new ones and that most development tools enable for automated and / or one click test case execution.
* JUnit provides also a graphical user interface (GUI) which makes it possible to write and test source code quickly and easily.
* JUnit shows test progress in a bar that is green if testing is going fine and it turns red when a test fails.

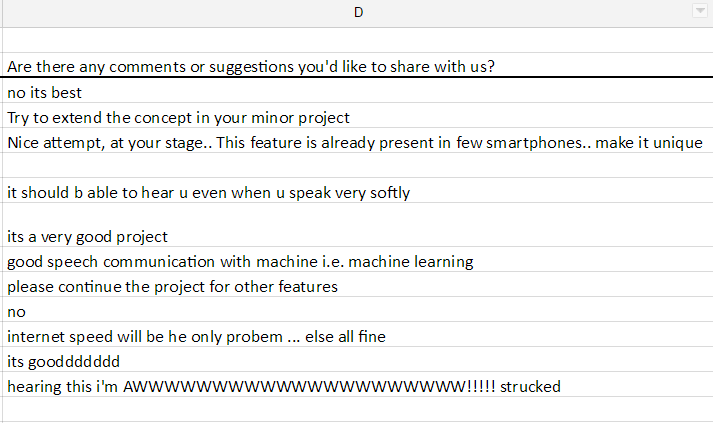
There is a lot of pleasure in seeing the green bars grow in the GUI output. A list of unsuccessful tests appears at the bottom of the display window. We can run multiple tests concurrently. The simplicity of JUnit makes it possible for the software developer to easily correct bugs as they are found.

**C. Feedback**

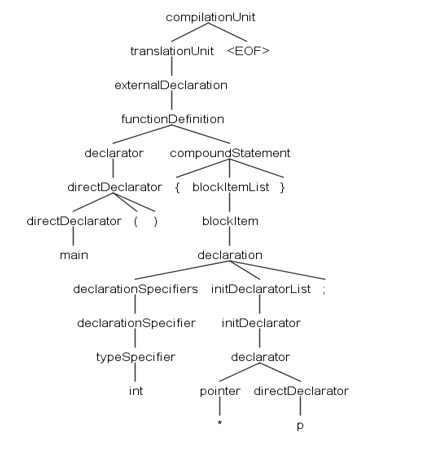






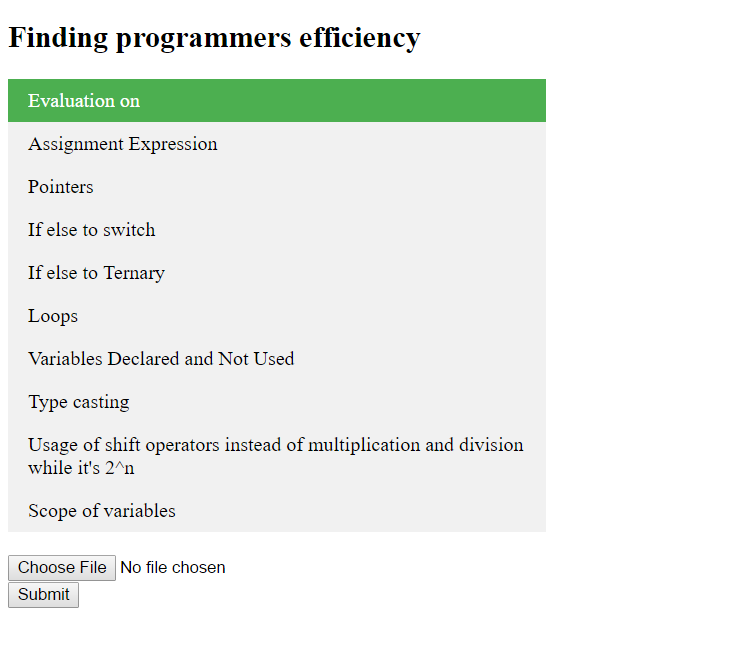


**D. Screenshots:**

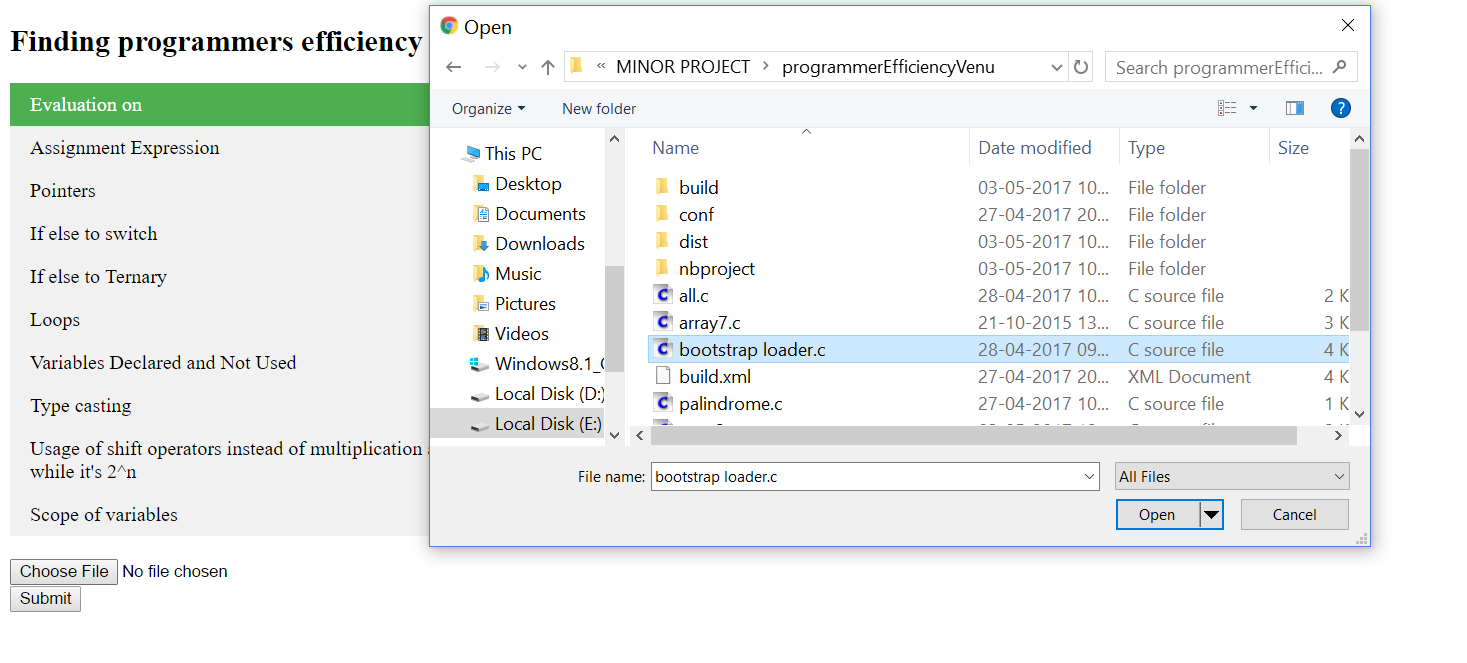


**Figure D.1**

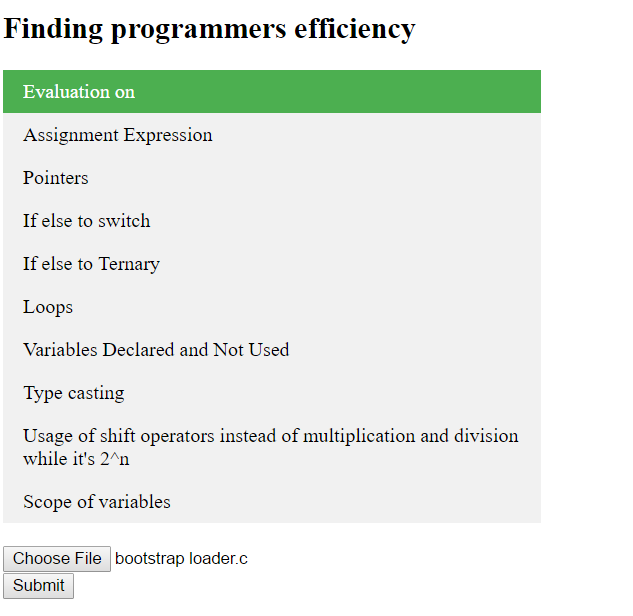
The tree generated for a simple pointer variable declaration in main().



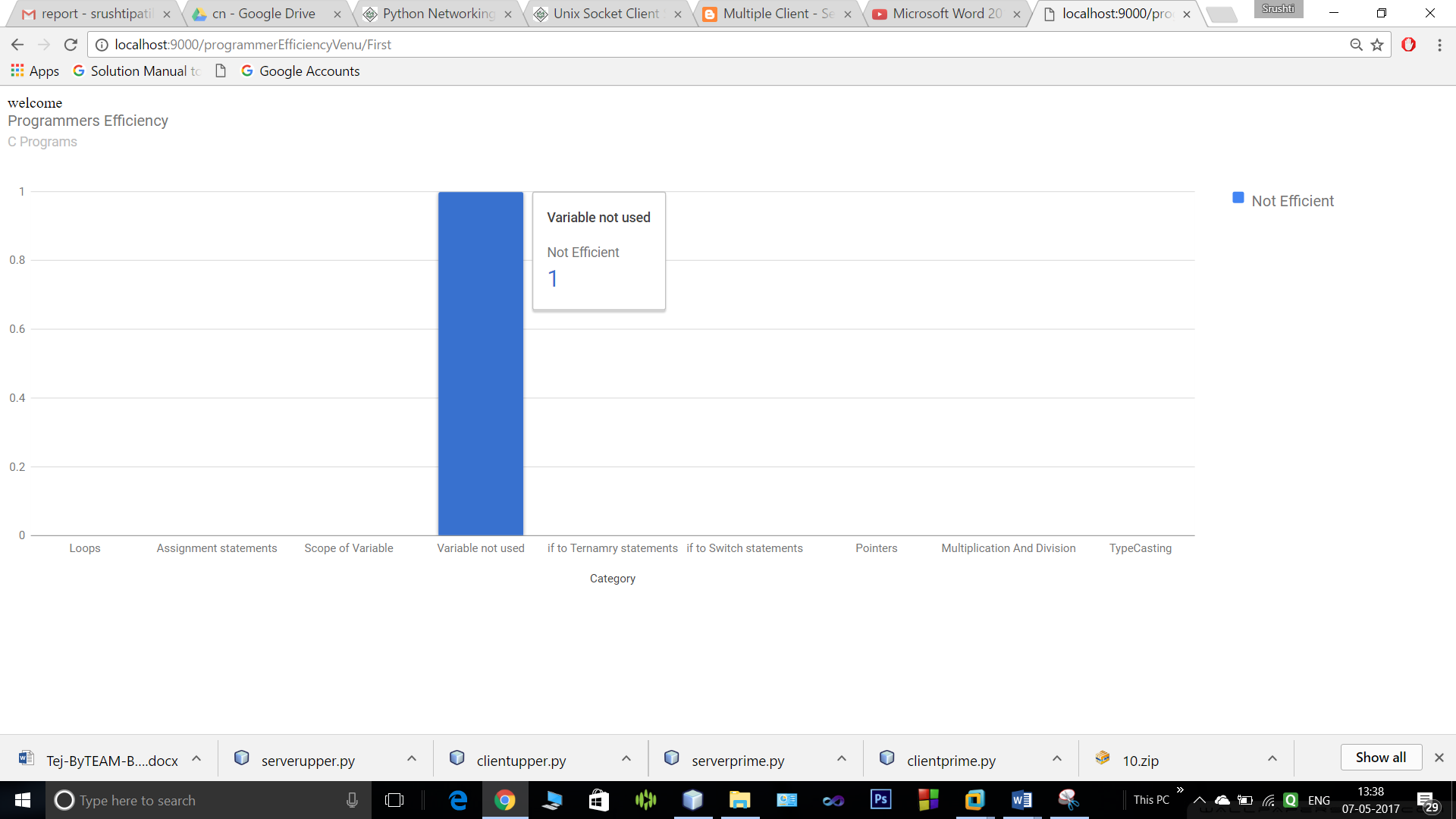
**Figure D.2**



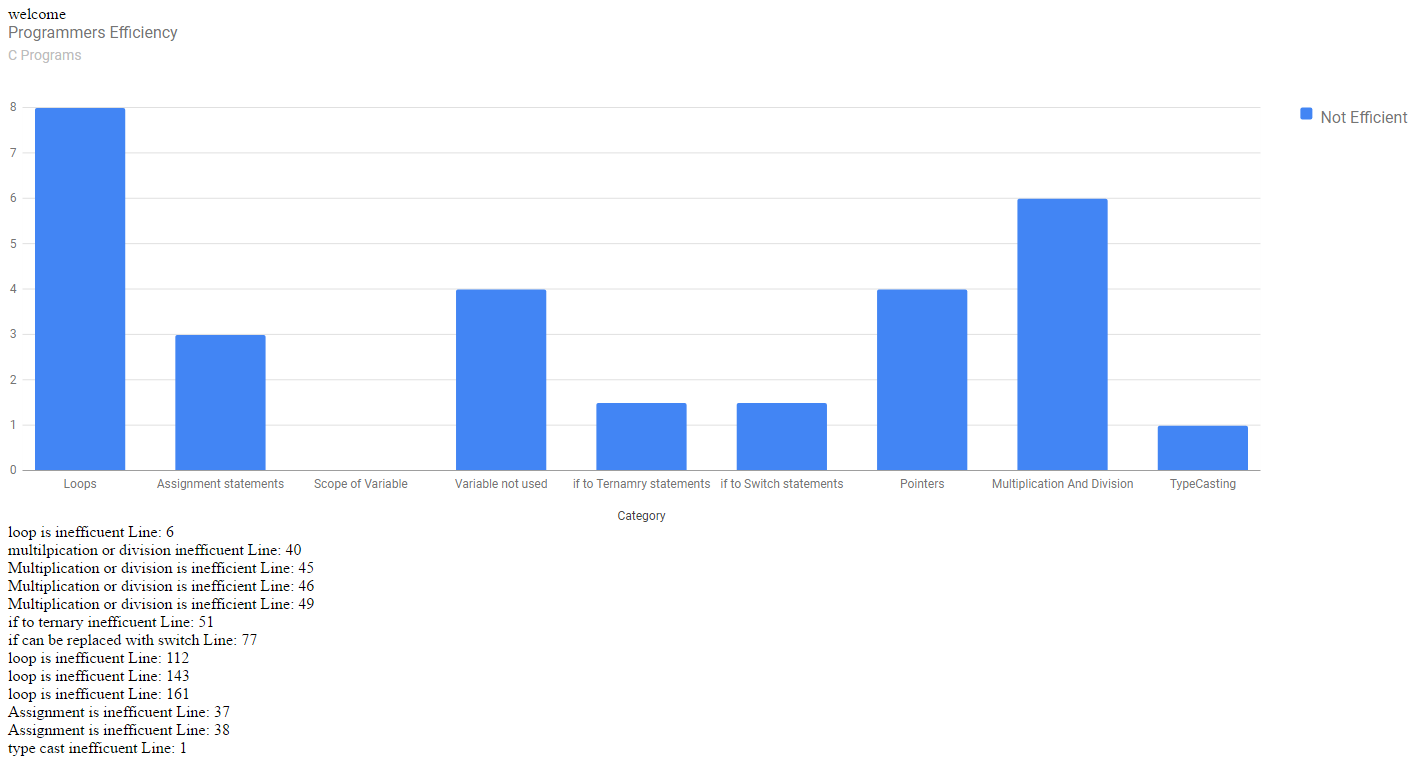
**Figure D.3**



**Figure D.4**

****

**Figure D.5**



**Figure D.6**