DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ST JOSEPH ENGINEERING COLLEGE, MANGALURU-575028

2019-2020

MINI PROJECT REPORT

ON

UNDO AND REDO OPERATION

DATA STRUCTURES & APPLICATIONS (18CS32)

Submitted by

1. Sana Parveen	4SO18CS099
2.Shravya	4SO18CS108
3.Siona Misha Nazareth	4SO18CS112
4. Sriganesh	4SO18CS113

Affiliated to Visvesvaraya Technological University, Belagavi

Table of Contents Pg no.

Chapter 1

Introduction

Today undo and redo feature has become an important aspect of a professional software, we can see those features in all most all the popular software. This is due to, ability to undo, go back or escape is a lifeline to many users. It is immensely reassuring to know that you can always undo something if you get it wrong and when that option is not available it can be incredibly disturbing and also redo plays major role in it.

This project introduces undo and redo features to a normal calculator program. This project has been developed in C Language. Proper comments are given at required location so has to make project user friendly.

Undo is a feature of a computer program that allows a user to cancel or reverse the last command executed. The redo function restores any action that have been previously undone using undo feature.

We have developed a code which can perform undo and redo operation, so that the user has more control on the output. It helps to correct the mistakes user has done while entering input. It increases the productivity as it decreases the time taken to rerun the code because of wrong inputs entered.

We have used concept of stacks to perform undo and redo operation. Stack is preferred for linear undo because stack follows LIFO that stores a history of all executed commands. When a new command is executed it is added to the top of stack. Therefore, only the last executed command can be undone and removed from the history. Undo can be repeated as long as the history is not empty. When undo feature is used it pops last execution and pushes that into separate stack (redo stack) . And when redo feature is used it pops the topmost execution into the original stack so as to neutralize the effect of undo

Chapter 2

Hardware and software requirements

Execution Platform :

Hardware requirements

Processor: Intel(R) Core(™) i5-8400 CPU @2.8Ghz Turbo upto 4.1GHz

RAM : 8GB DD4 2400MHz

Hard disk capacity: 1 TB HDD,

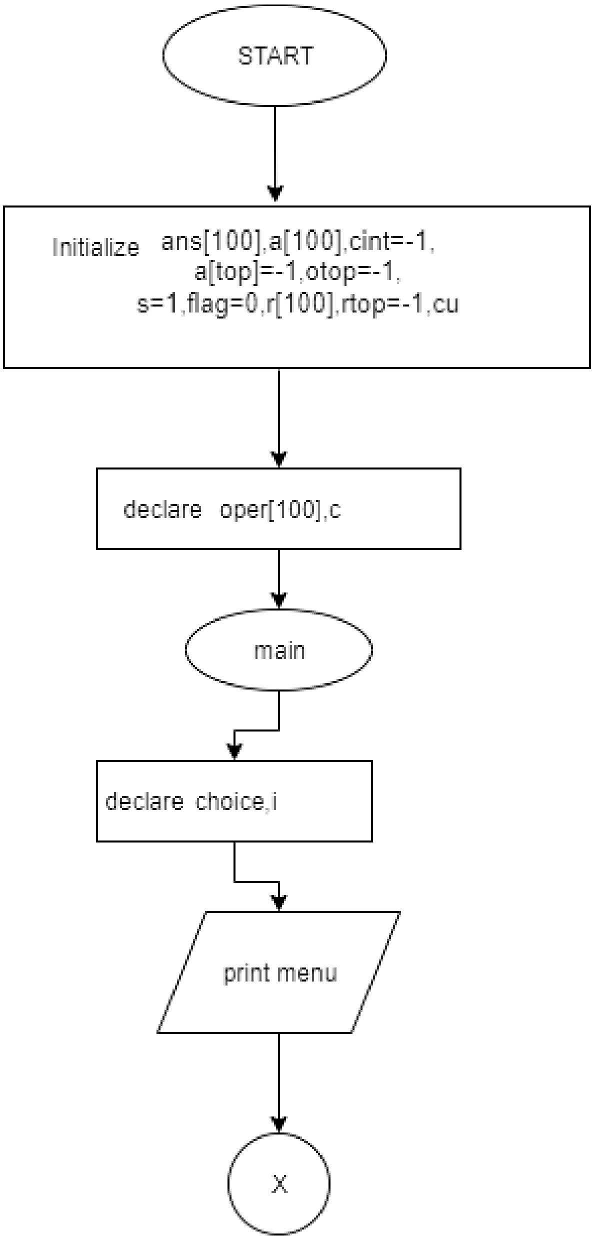
Software requirements

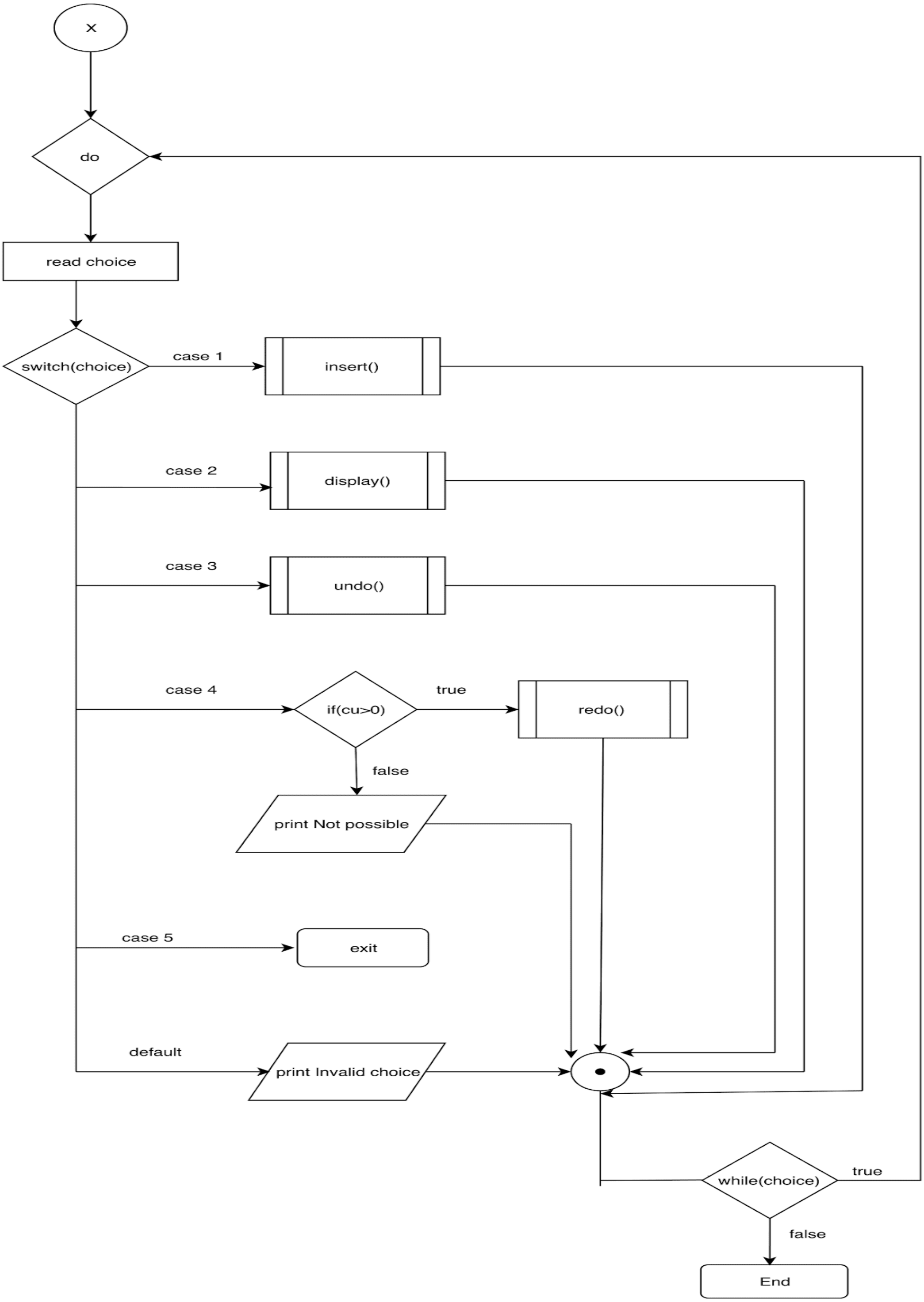
Operating System:	Windows 10 Pro
Compiler:	TDM-GCC MinGW Compiler 5.10.2
IDE :	Atom 1.41.0

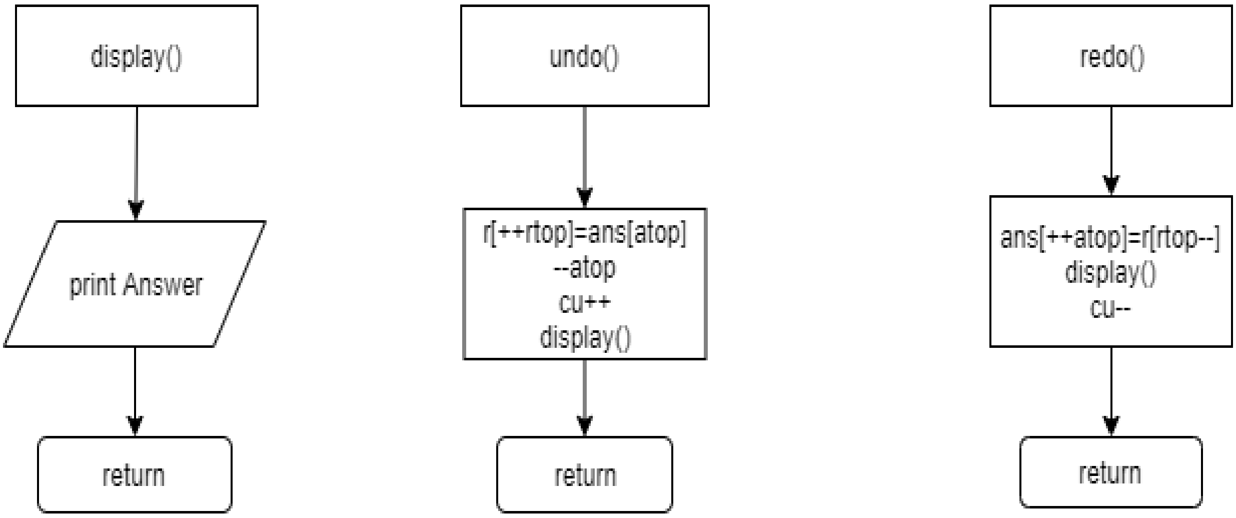
Chapter 3

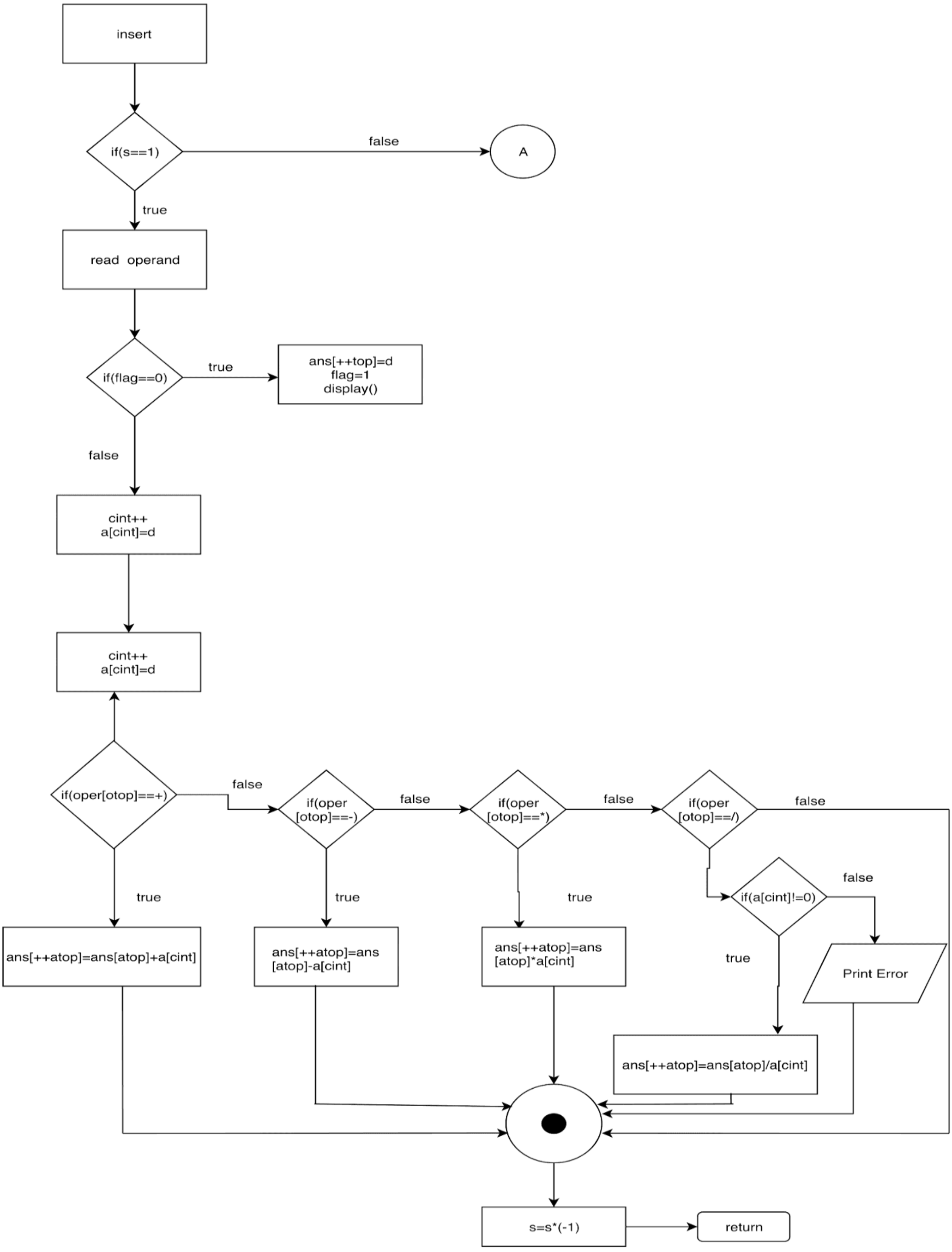
Design

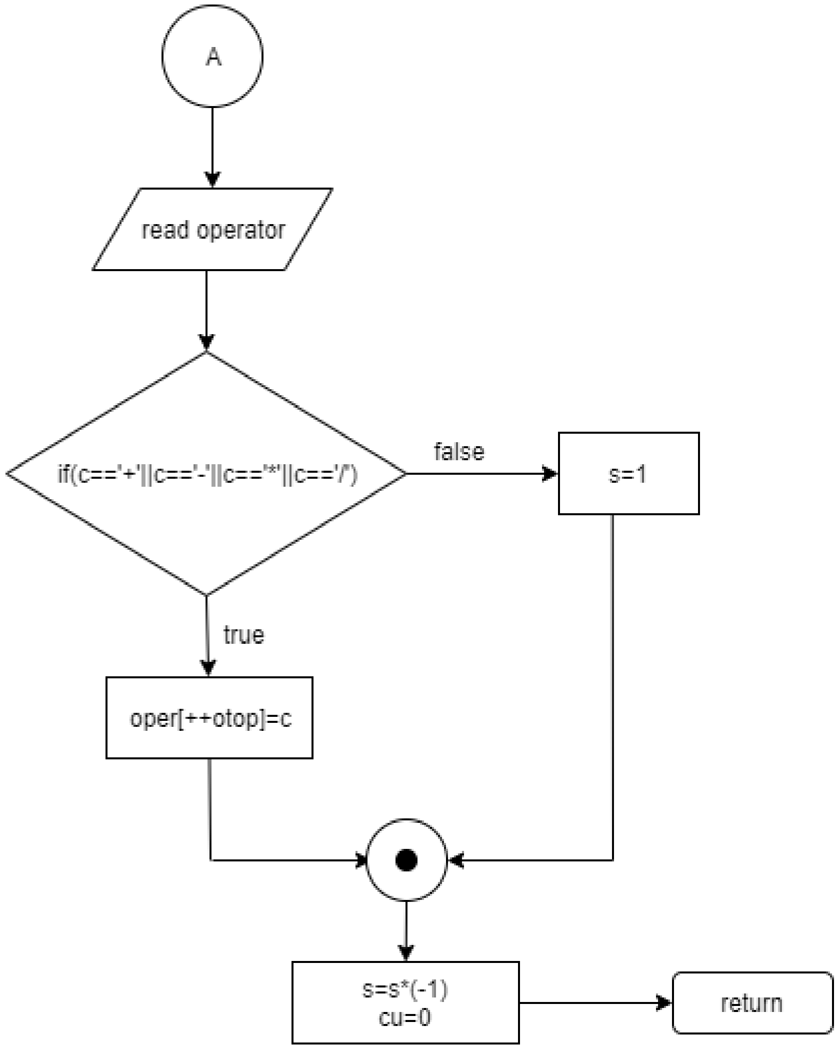
Flowchart




×
do
read choice
switch(choice)
case 1
insert()
case 2
display()
case 3
undo()
case 4
if(cu>0)
true
redo()
false
print Not possible
case 5
exit
default
print Invalid choice
while(choice)
true
false
End


display()
print Answer
return
undo()
r[++rtop]=ans[atop]
--atop
cu++
display()
return
redo()
ans[++atop]=r[rtop--]
display()
cu--
return


insert
if(s==1)
false
A
true
read operand
if(flag==0)
true
ans[++top]=d
flag=1
display()
false
cint++
a[cint]=d
cint++
a[cint]=d
if(oper[otop]==+)
false
if(oper
[otop]==-)
false
if(oper
[otop]==*)
false
if(oper
[otop]==/)
false
true
true
true
if(a[cint]!=0)
false
ans[++atop]=ans[atop]+a[cint]
ans[++atop]=ans
[atop]-a[cint]
ans[++atop]=ans
[atop]*a[cint]
true
Print Error
ans[++atop]=ans[atop]/a[cint]
s=s*(-1)
return


A
read operator
if(c=='+'||c=='-'||c=='*'||c=='/')
false
s=1
true
oper[++otop]=c
s=s*(-1)
cu=0
return

Algorithm

```
//main function
 //initialize and declare the members  ans ,a ,cint←-1,atop← -1,otop← -1,s←1,flag← 0 ,rtop←1,cu.
Step 1: Print ”MENU”
        1.insert  2.display  3.undo  4.redo  5.exit
        Switch(choice)
                Case1: insert()
                        Goto  step3.
                Case2: display()
                        Goto step 3.
                Case3: undo()
                        Goto step 3.
                Case 4: if(cu>0)     do
                         redo()                //undo >=redo  No. of redo cannot exceed no
                        else                   // of undo
                             Print “ Redo is not possible”.
                        Endif.
                        Goto step 3.
                Case 5: exit()
Step 2: while(choice)
Step 3:End.
```

```
//function to insert the elements.

Step 1: if(s=1)                        //s is used to insert an operator after operand or
                                       // vice versa
      //Read the operand from the user.

Step 2: if( flag=0)     do
            ans[++atop]← d;     //first operand is pushed directly to the ans stack
            flag←1;                 //atop is stack pointer for ans stack
            //display()
      else
            cint++;            //else it is inserted to  operand stack pointed by cint
            a[cint]← d;
      endif

Step 3: if(oper[atop]=’+’)    do
            ans[++atop]← ans[atop]+a[cint] ;
      else if(oper[otop]=’-‘)    do                //calculation is performed as per the
             ans[++atop]=←ans[atop]-a[cint];       //operators
      else if(oepr[otop]==’*’)    do               //ans is pushed into the ans stack
            ans[++atop]←ans[otop]*a[cint];
      else if(oper[otop]==’/’)   do{
             if(a[cint]!=0)                          //Divide by zero
             ans[++atop]←ans[atop]+a[cint];

  else
         print Divide by zero
          }
```

```
        endif.


Step 4: s←s*(-1).
//if step 1 is false ,  read the operator form the user.


Step 5: if(c ='+' || c ='-' || c = '*' || c =='/')      do
                oper[++otop]← c;                       //valid operator are pushed into
                                                        //operator stack
        else                                            //which is pointed by otop
                print " Invalid"
                s←1;
        end if
Step 6: s← s*(-1)

Step 7: cu←0;

Step 8: End
```

```
//function to display the contents
Step 1: print "Answer"
        //answer is stored in ans[atop]
Step 2: End

//function to perform undo operation
Step 1:  r[++rtop]← ans[atop]          //popped from ans and pushed into redo stack
Step 2: --atop                          //atop now points to previous answer
Step 3: cu++
Step 4: display()                       //function to display elements
Step 5: end

//function to perform redo operation
Step 1: ans[++atop]←r[top--]     //popped from redo stack and pushed  into ans
stack
Step 2: display()                //function to display elements
Step 3: cu--
Step 4: End
```

Chapter 4

Implementation

```
/*


  _____   _____ _____           _           _
 |  __ \ / ____|  __ \         (_)         | |
 | |  | | (___ | |__) | __ ___  _  ___  ___| |_
 | |  | |\___ \|  ___/ '__/ _ \| |/ _ \/ __| __|
 | |__| |____) | |   | | | (_) | |  __/ (__| |_
 |_____/|_____/|_|   |_|  \___/| |\___|\___|\__|
                              _/ |
                             |__/


*/
#include<stdio.h>                      //Required comments are provided and
#include<stdlib.h>                      // explained in algorithm

float a[100],r[100],ans[100];
int cint=-1,atop=-1,otop=-1,s=1,flag=0,rtop=-1,cu=0;
char oper[100],c;
void undo();
void insert();
void display() ;
void redo() ;





int main()
```

```
{
    int i,choice;

printf("_________________MENU________________\n\n\t\t1.Insert\n\t\t2.Display\
n\t\t3.Undo\n\t\t4.Redo\n\t\t5.Exit\n");
      printf("\n_____________________________________\n");
    do{
        printf("\nEnter your choice :\t");
        scanf("%d",&choice);

    switch(choice)
    {
        case 1: insert();
            break;
        case 3: undo() ;

                break;
        case 2:display();
                break;
        case 4:if(cu>0)
                redo();
               else
                printf("\nNot Possible\n");


                   break;
        case 5:exit(0);
        default :  printf("Oops!! It seems you have entered Invalid choice\n");



    }}while(choice);

 }
```

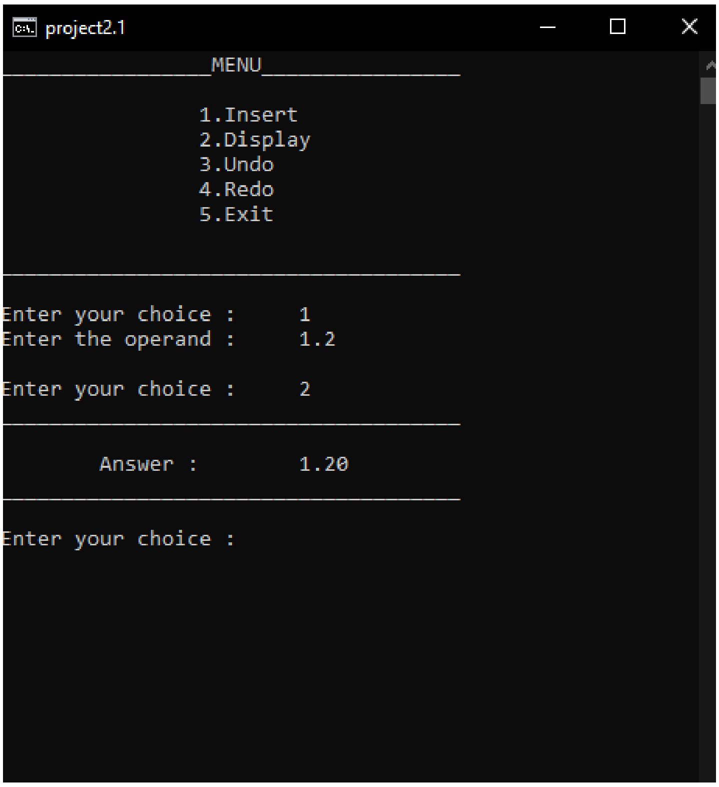
```
void insert()
{
  float d;
  cu=0;

   if(s==1)
    {



     printf("Enter the operand :\t");
     scanf("%f",&d);
     if(flag==0)
     {
         ans[++atop]=d;
         flag=1;

     }
     else
     {
         cint++;
         a[cint]=d;

     }
     if(oper[otop]=='+')
        ans[++atop]=ans[atop]+a[cint];

    else if(oper[otop]=='-')
            ans[++atop]=ans[atop]-a[cint];

     else if(oper[otop]=='*')
            ans[++atop]=ans[atop]*a[cint];

      else if(oper[otop]=='/')
             {  if(a[cint]!=0)
```

```
                ans[++atop]=ans[atop]/a[cint];
              else
                printf("\nError!!Divide by Zero\n");
             }

  s=s*(-1);
   }
   else
  {
     printf("Enter the operator:\t");
     scanf("%s",&c);
    if(c=='+'||c=='-'||c=='*'||c=='/')
    {
       oper[++otop]=c;

    }
    else
   {
     printf("Oops!! It seems you have entered Invalid opertor\n");
       s=1;
   }
    s=s*(-1);
 }
}

void display()
{
printf("_______________________________________\n\n");
   printf("\tAnswer :\t%.2f\n",ans[atop]);
   printf("_______________________________________\n");
}

void undo()
{
  r[++rtop]=ans[atop];
  --atop;
   cu+=1;
   printf("\nUndo is Successful\n");
  display();
}

void redo()
{
  --cu;
  ans[++atop]=r[rtop--];
  printf("\nRedo is Successful\n");
  display();
}
```

Chapter 5

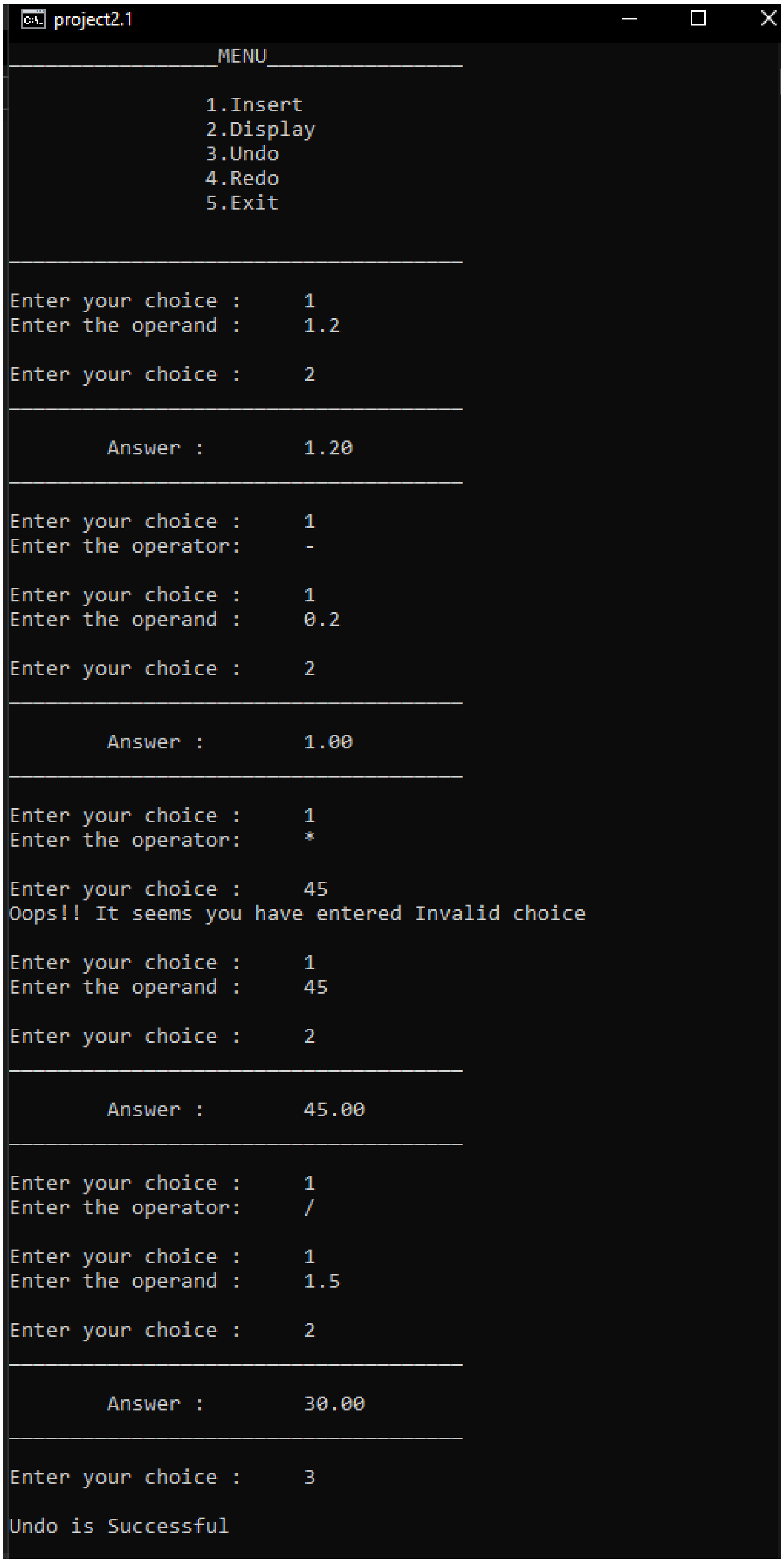
Results

CASE 1 : When only one oparand is entered.



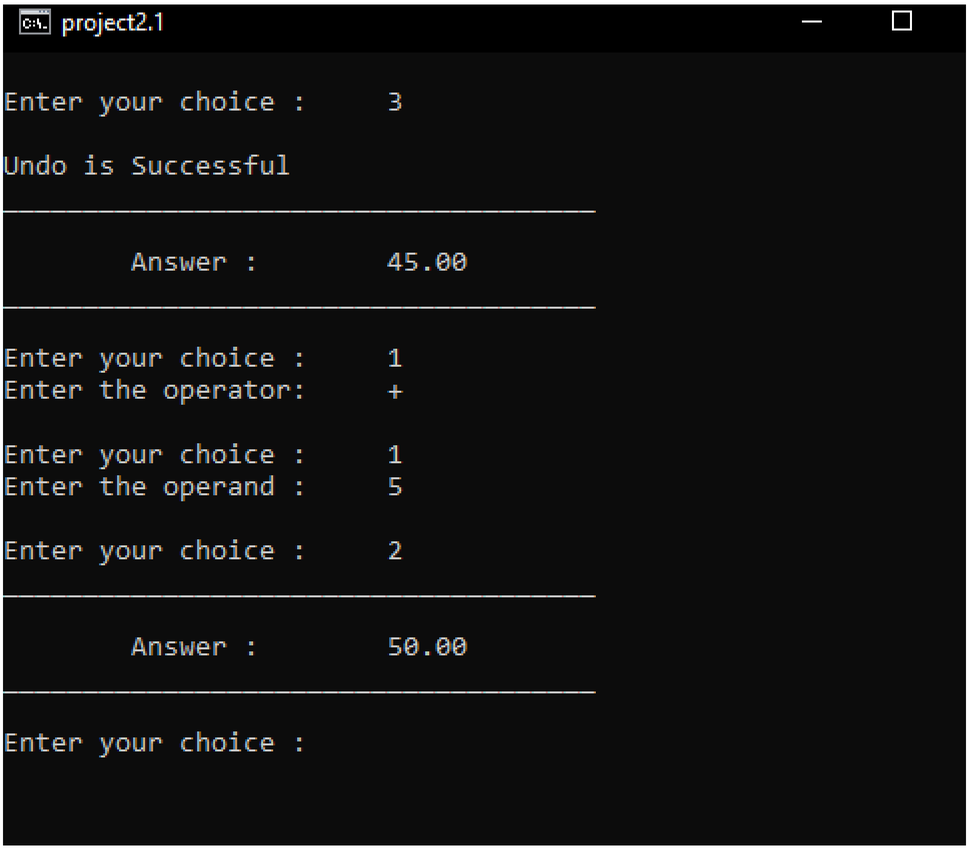
```
_________________MENU_________________

                1.Insert
                2.Display
                3.Undo
                4.Redo
                5.Exit


______________________________________

Enter your choice :      1
Enter the operand :      1.2

Enter your choice :      2
______________________________________

        Answer :         1.20

______________________________________

Enter your choice :
```

CASE 2 : All operations and single undo



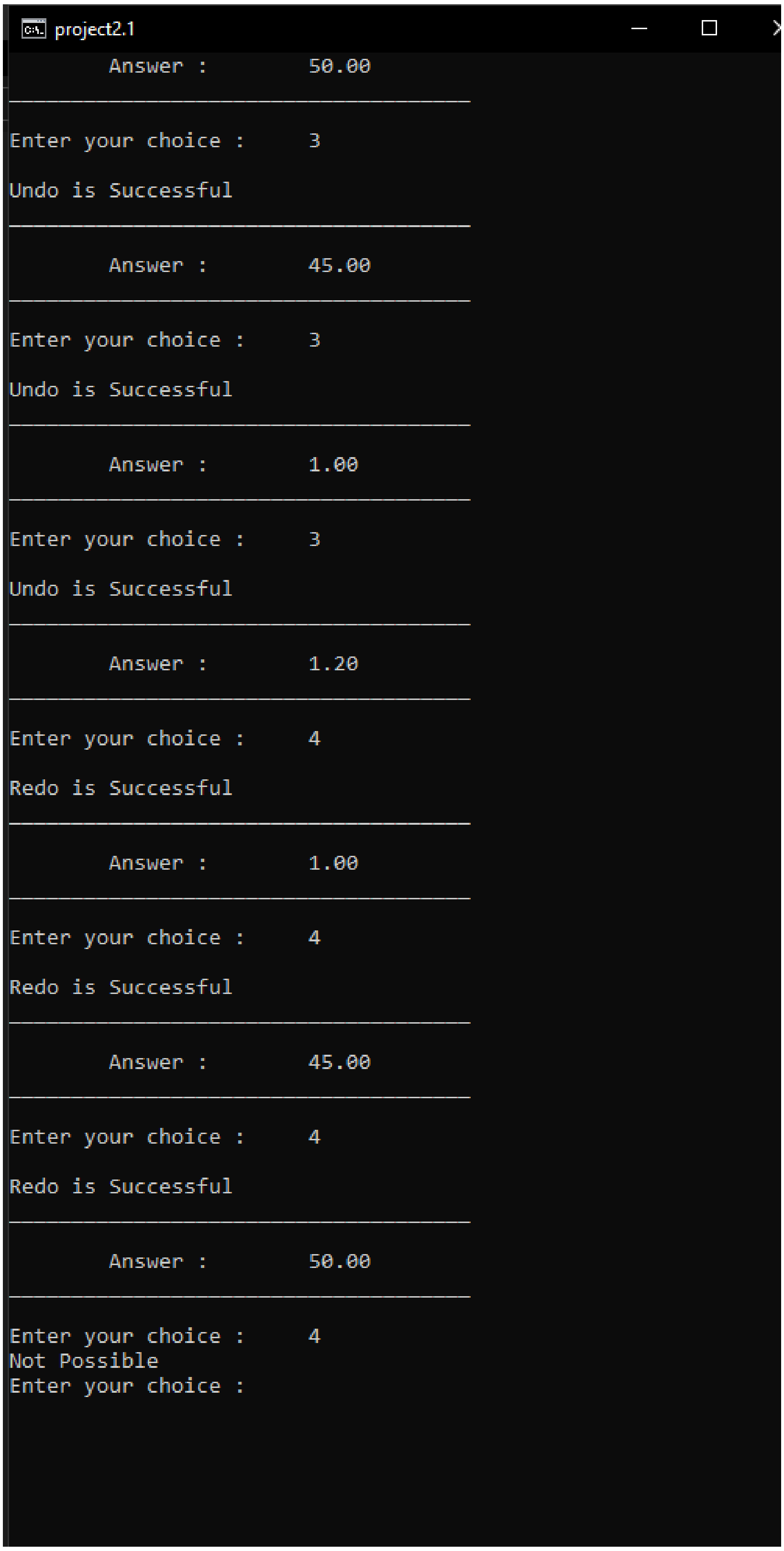
```
_________________MENU________________

                1.Insert
                2.Display
                3.Undo
                4.Redo
                5.Exit

_____________________________________

Enter your choice :     1
Enter the operand :     1.2

Enter your choice :     2
_____________________________________

        Answer :        1.20
_____________________________________

Enter your choice :     1
Enter the operator:     -

Enter your choice :     1
Enter the operand :     0.2

Enter your choice :     2
_____________________________________

        Answer :        1.00
_____________________________________

Enter your choice :     1
Enter the operator:     *

Enter your choice :     45
Oops!! It seems you have entered Invalid choice

Enter your choice :     1
Enter the operand :     45

Enter your choice :     2
_____________________________________

        Answer :        45.00
_____________________________________

Enter your choice :     1
Enter the operator:     /

Enter your choice :     1
Enter the operand :     1.5

Enter your choice :     2
_____________________________________

        Answer :        30.00
_____________________________________

Enter your choice :     3

Undo is Successful
```

CASE 3 : continuation after performing successful undo



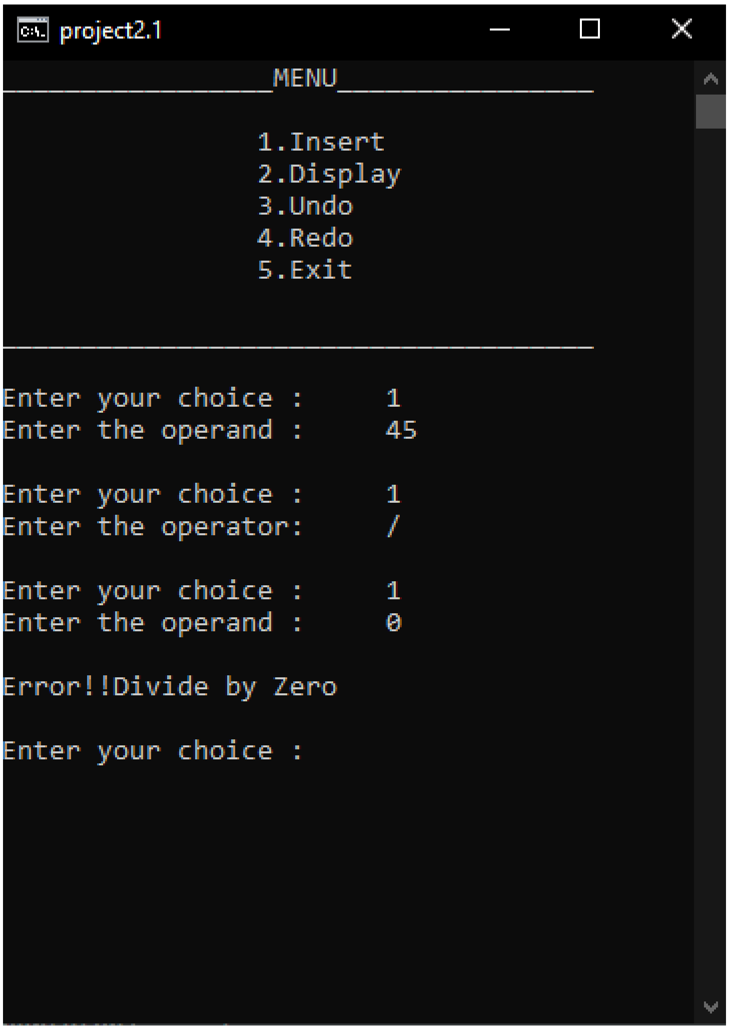
```
Enter your choice :     3

Undo is Successful
_____________________________________

        Answer :        45.00
_____________________________________

Enter your choice :     1
Enter the operator:     +

Enter your choice :     1
Enter the operand :     5

Enter your choice :     2
_____________________________________

        Answer :        50.00
_____________________________________

Enter your choice :
```

CASE 4: Performing Undo & Redo until not possible condition



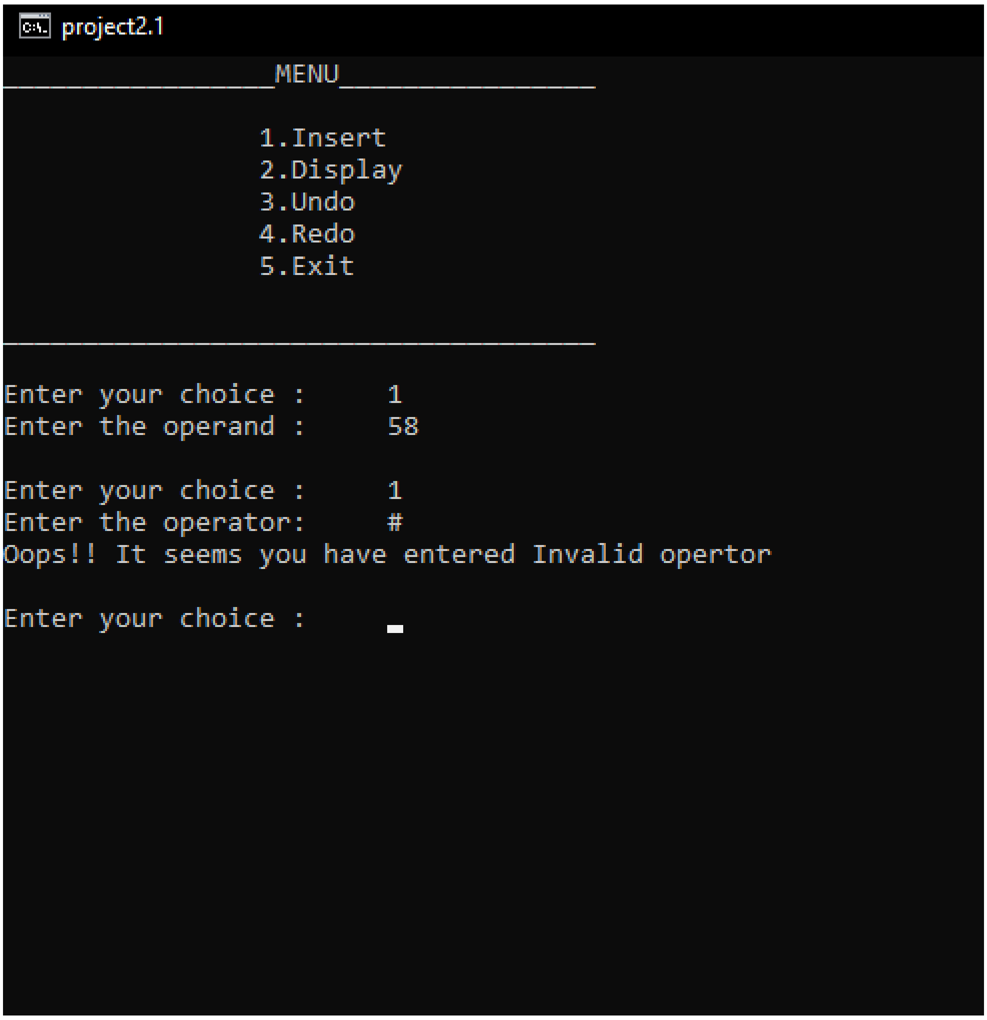
```
        Answer :        50.00
_____________________________________

Enter your choice :     3

Undo is Successful
_____________________________________

        Answer :        45.00
_____________________________________

Enter your choice :     3

Undo is Successful
_____________________________________

        Answer :        1.00
_____________________________________

Enter your choice :     3

Undo is Successful
_____________________________________

        Answer :        1.20
_____________________________________

Enter your choice :     4

Redo is Successful
_____________________________________

        Answer :        1.00
_____________________________________

Enter your choice :     4

Redo is Successful
_____________________________________

        Answer :        45.00
_____________________________________

Enter your choice :     4

Redo is Successful
_____________________________________

        Answer :        50.00
_____________________________________

Enter your choice :     4
Not Possible
Enter your choice :
```

CASE 5 : Divide By 0



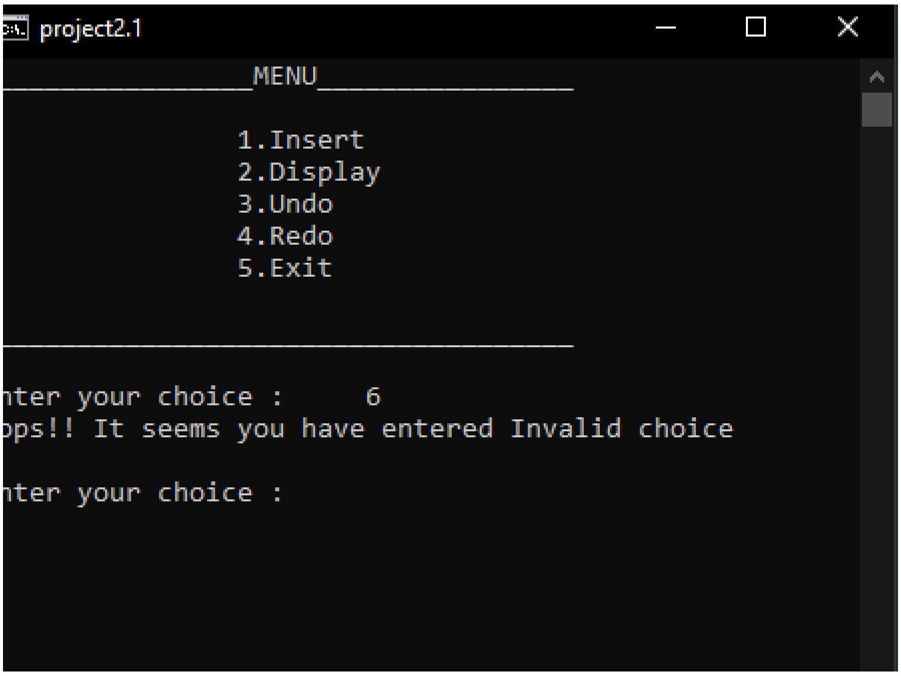
```
_________________MENU________________

                1.Insert
                2.Display
                3.Undo
                4.Redo
                5.Exit


_____________________________________

Enter your choice :     1
Enter the operand :     45

Enter your choice :     1
Enter the operator:     /

Enter your choice :     1
Enter the operand :     0

Error!!Divide by Zero

Enter your choice :
```

CASE 6: Invalid Operator



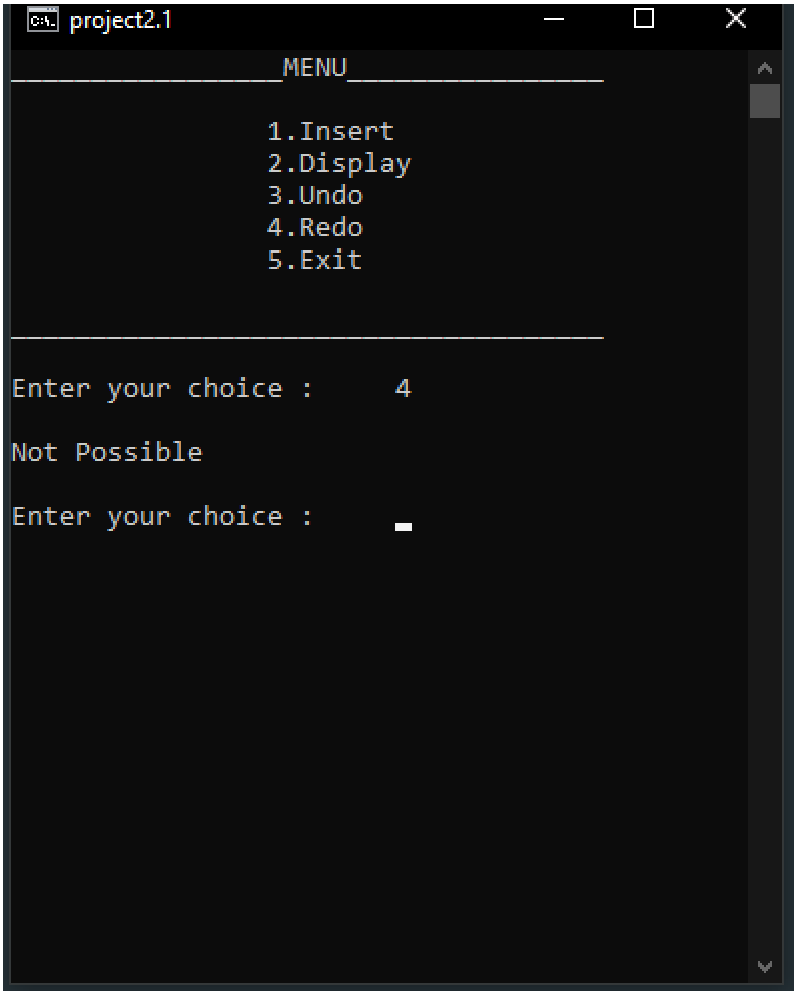
```
_________________MENU________________

                1.Insert
                2.Display
                3.Undo
                4.Redo
                5.Exit


_____________________________________

Enter your choice :     1
Enter the operand :     58

Enter your choice :     1
Enter the operator:     #
Oops!! It seems you have entered Invalid opertor

Enter your choice :     _
```

CASE 7 : Invalid Choice



```
________________MENU________________

               1.Insert
               2.Display
               3.Undo
               4.Redo
               5.Exit


____________________________________

nter your choice :     6
ops!! It seems you have entered Invalid choice

nter your choice :
```

CASE 8 : Not possible redo condition.



```
_________________MENU________________

                1.Insert
                2.Display
                3.Undo
                4.Redo
                5.Exit


_____________________________________

Enter your choice :      4

Not Possible

Enter your choice :
```

References

1. Ellis Horowitz and Sartaj Sahni, Fundamentals of Data Structures in C, 2nd Ed, Universities Press,2014.
2. Seymour Lipschutz, Data Structures Schaum's Outlines, Revised 1st Ed, McGraw Hill, 2014.
3. Reema Thareja, Data Structures using C, 3rd Ed, Oxford press, 2012.
4. Jean-Paul Tremblay & Paul G. Sorenson, An Introduction to Data Structures with Applications, 2nd Ed, McGraw Hill, 2013.
5. A M Tenenbaum, Data Structures using C, PHI, 1989
6. Robert Kruse, Data Structures and Program Design in C, 2nd Ed, PHI, 1996.

*This project is developed from scratch