M.Sc. (Five Year Integrated) in Computer Science (Artificial Intelligence & Data Science)

Second Semester

Laboratory Record

21-805-0206: DATA STRUCTURES LAB

Submitted in partial fulfillment
of the requirements for the award of degree in
Master of Science (Five Year Integrated)
in Computer Science (Artificial Intelligence & Data Science) of
Cochin University of Science and Technology (CUSAT)
Kochi

Submitted by

OMAL S.
(80521015)

DEPARTMENT OF COMPUTER SCIENCE
COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY (CUSAT)
KOCHI-682022

SEPTEMBER 2022

DEPARTMENT OF COMPUTER SCIENCE

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY (CUSAT)

KOCHI, KERALA-682022

*This is to certify that the software laboratory record for **21-805-0206: Data Structures Lab** is a record of work carried out by **OMAL S (80521015)**, in partial fulfillment of the requirements for the award of degree in **Master of Science (Five Year Integrated)** in **Computer Science (Artificial Intelligence & Data Science)** of Cochin University of Science and Technology (CUSAT), Kochi. The lab record has been approved as it satisfies the academic requirements in respect of the second semester laboratory prescribed for the Master of Science (Five Year Integrated) in Computer Science degree.*

Faculty Member in-charge

Dr. Madhu S. Nair	Dr. Philip Samuel
Professor	Professor and Head
Department of Computer Science	Department of Computer Science
CUSAT	CUSAT

Table of Contents

INSERTION AND DELETION IN AN ARRAY

AIM

To understand the insertion and deletion of elements in the array using functions such as INSERT() and DELETE()

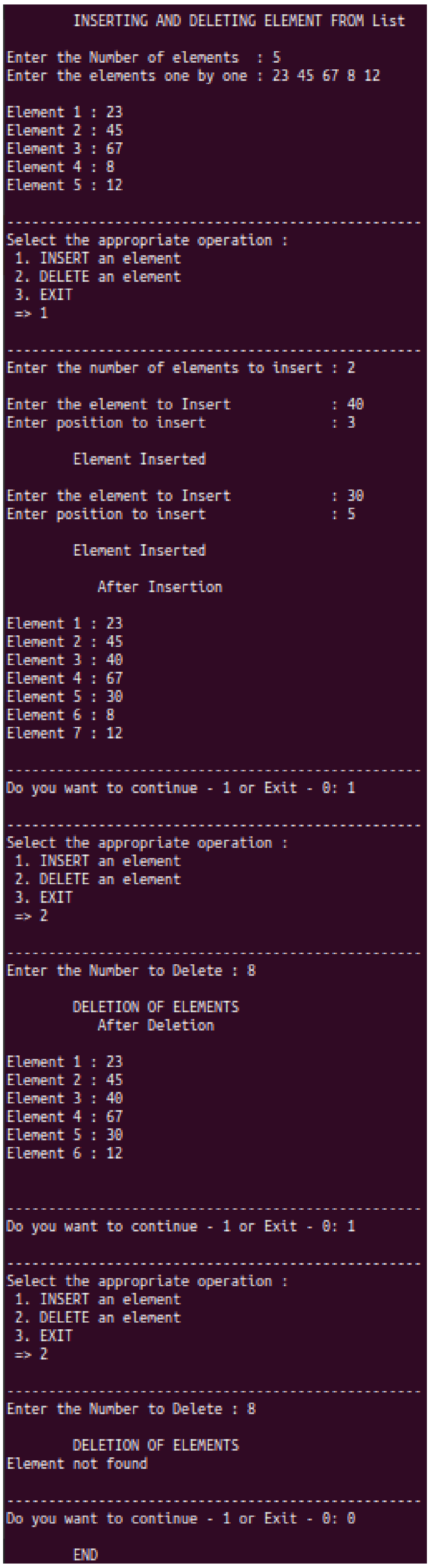
ALGORITHM

```
    INSERT( DATA , N , POS , ITEM )
       1. set  J = N
       2. Repeat the steps (i) & (ii) while J >= POS
             (i)  DATA[J] = DATA[J-1]
             (ii) J = J-1
       3. end while loop
       4. DATA[POS] = ITEM
       5. N = N+1
       6. Exit

    DELETE(DATA , N ,ITEM )
       1. set J = 1
       2. Repeat the step (i) for J=1 to N-1
           (i)  if ITEM = DATA[J]
                    (a) Repeat the step for K=J  to K<N
                           DATA[K] = DATA[K+1]
                    (b) end for loop
           (ii)  end if condition
       3. end for loop
       4. set N = N-1
       5. Exit
```

PROGRAM

```
#include<iostream>
using namespace std;

void INSERT(int Temp_List[],int Temp_Number_of_terms,int Temp_position,\\
                  int Temp_Value)
{
  cout<<"\n\tElement Inserted\n";

  while (Temp_Number_of_terms >= Temp_position)
  {
    Temp_List[Temp_Number_of_terms] = Temp_List[Temp_Number_of_terms -1];
    Temp_Number_of_terms = Temp_Number_of_terms - 1;
  }
  Temp_List[Temp_position] = Temp_Value;
}

int DELETE(int Temp_List[],int Temp_Number_of_terms,int Temp_Value)
{
 cout<<"\n\tDELETION OF ELEMENTS";

 for (int index = 0 ; index<Temp_Number_of_terms ; index++)
 {
   if(Temp_Value == Temp_List[index])
   {
     for (int Number = index; Number< Temp_Number_of_terms;Number++)
     {
       Temp_List[Number] = Temp_List[Number+1];
     }
     return 1;
   }
}
return 0;
}

int main()
{
 int Number_of_terms, Value , Position , index , choice , option,terms;
 cout<<"\n\tINSERTING AND DELETING ELEMENT FROM List\n\n";
 cout<<"Enter the Number of elements  : ";
 cin>>Number_of_terms;
 int List[Number_of_terms];
 cout<<"Enter the elements one by one : ";

 for (index=0;index<Number_of_terms;index++)
 {
  cin>>List[index];
 }
 cout<<endl;
 for (index=0;index<Number_of_terms;index++)
 {
  cout<<"Element "<<index+1<<" : "<<List[index]<<endl;
 }

 do{
   cout<<"\n--------------------------------------------------";
   cout<<"\nSelect the appropriate operation : \n 1. INSERT an element \n \\
            2. DELETE an element \n 3. EXIT \n => ";
   cin>>choice;
   cout<<"\n--------------------------------------------------";

 switch(choice)
 {
   case 1:
    { cout<<"\nEnter the number of elements to insert : ";
      cin>>terms;
      List[Number_of_terms+terms];
      for(int i=0;i<terms;i++)
      {
         cout<<"\nEnter the element to Insert            : ";
         cin>>Value;
         cout<<"Enter position to insert               : ";
         Number_of_terms = Number_of_terms+1;
         cin>>Position;
         Position = Position-1;
         INSERT(List,Number_of_terms,Position,Value);
      }
      cout<<"\n\t   After Insertion \n\n";
      for ( index=0;index<Number_of_terms;index++)
      {
          cout<<"Element "<<index+1<<" : "<<List[index]<<endl;
      }
      break; }

    case 2:
      {
        if(Number_of_terms != 0)
        {
         cout<<"\nEnter the Number to Delete : ";
         cin>>Value;
 if (DELETE(List,Number_of_terms,Value) == 0)
 {
   cout<<"\nElement not found\n";
 }
         else
         {
     cout<<"\n\t   After Deletion \n\n";
   for ( index=0;index<Number_of_terms-1;index++)
   {
   cout<<"Element "<<index+1<<" : "<<List[index]<<endl;
           }
     cout<<endl;
     Number_of_terms = Number_of_terms-1;
    }
   }
   else
   {
     cout<<"\nUNDERFLOW No elements in the array to delete\n";
   }
    break;
  }

    case 3:
          break;
    default: cout<<"\nWrong Choice\n";
  }

    if (choice != 3)
    {
      cout<<"\n--------------------------------------------------";
      cout<<"\nDo you want to continue - 1 or Exit - 0: ";
      cin>>option;
      if(option == 0)
      {
        choice = 3;
      }
    }

 }while(choice != 3);
 cout<<"\n\tEND\n";
 return 0;
}
```

SAMPLE INPUT-OUTPUT



```
        INSERTING AND DELETING ELEMENT FROM List

Enter the Number of elements   : 5
Enter the elements one by one : 23 45 67 8 12

Element 1 : 23
Element 2 : 45
Element 3 : 67
Element 4 : 8
Element 5 : 12

--------------------------------------------------
Select the appropriate operation :
 1. INSERT an element
 2. DELETE an element
 3. EXIT
 => 1

--------------------------------------------------
Enter the number of elements to insert : 2

Enter the element to Insert            : 40
Enter position to insert               : 3

        Element Inserted

Enter the element to Insert            : 30
Enter position to insert               : 5

        Element Inserted

           After Insertion

Element 1 : 23
Element 2 : 45
Element 3 : 40
Element 4 : 67
Element 5 : 30
Element 6 : 8
Element 7 : 12

--------------------------------------------------
Do you want to continue - 1 or Exit - 0: 1

--------------------------------------------------
Select the appropriate operation :
 1. INSERT an element
 2. DELETE an element
 3. EXIT
 => 2

--------------------------------------------------
Enter the Number to Delete : 8

        DELETION OF ELEMENTS
           After Deletion

Element 1 : 23
Element 2 : 45
Element 3 : 40
Element 4 : 67
Element 5 : 30
Element 6 : 12


--------------------------------------------------
Do you want to continue - 1 or Exit - 0: 1

--------------------------------------------------
Select the appropriate operation :
 1. INSERT an element
 2. DELETE an element
 3. EXIT
 => 2

--------------------------------------------------
Enter the Number to Delete : 8

        DELETION OF ELEMENTS
Element not found

--------------------------------------------------
Do you want to continue - 1 or Exit - 0: 0

        END
```

LINEAR SEARCH

AIM

To under the searching of a number using the Linear search technique. To identify the comparison of elements linearly to locate the search item.

ALGORITHM

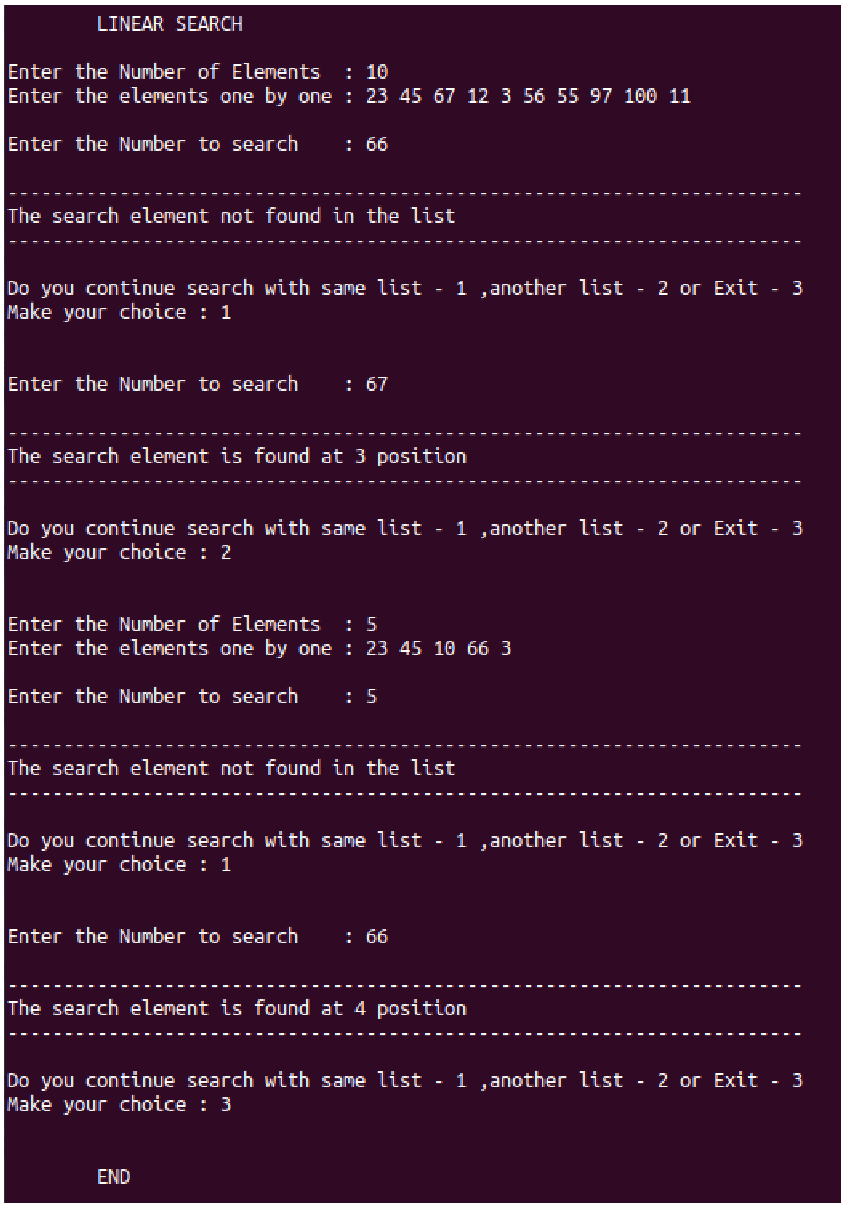
```
    LINEAR_SEARCH( DATA, N, ITEM)
        1.  set INDEX = 1, POS = NULL
        2.  Repeat the steps (i) & (ii) while INDEX<N and POS == NULL
            (i)     if DATA[INDEX] = ITEM
                        (a) POS = INDEX
                        (b) BREAK the condition
            (ii)    end if condition
            (iii)   INDEX = INDEX+1
        3.  end while loop
        4.  if POS not equal to NULL
                (i) print ITEM located at POS.
        5.  else
                (i) print ITEM not located.
        6.  end if condition
        7.  Exit
```

PROGRAM

```
#include<iostream>
#include<iomanip>
using namespace std;
void LINEAR_SEARCH(int List_Data[],int Terms,int search_key)
{
 int index = 0, Position = (-1);
 while (index < Terms & Position == (-1))
 {
  if(List_Data[index] == search_key)
  {
   Position = index;
   break;
  }
  index++;
 }
 if(Position != -1)
```

```
     cout<<"\nThe search element is found at "<<index+1<<" position";
 else
     cout<<"\nThe search element not found in the list";
}

int main()
{
 cout<<"\n\tLINEAR SEARCH\n";
 int Number_of_terms,search_element,choice = 1,option=6, position;

 do{
 cout<<"\nEnter the Number of Elements  : ";
 cin>>Number_of_terms;
 int List[Number_of_terms];



 cout<<"Enter the elements one by one : ";
 for (int loop_variable = 0;loop_variable < Number_of_terms;loop_variable++)
 {
  cin>>List[loop_variable];
 }

 do {
 cout<<"\nEnter the Number to search    : ";
 cin>>search_element;

 cout<<"\n--------------------------------------------"//
                    "-------------------------";

 LINEAR_SEARCH(List,Number_of_terms,search_element);

 cout<<"\n------------------------------------------------"//
                   "----------------------";
 cout<<endl;

 cout<<"\nDo you continue search with same list - 1 ,another list - 2"//
       "or Exit - 3 \nMake your choice : ";
 cin>>choice;
 cout<<"\n";
 }while(choice == 1 ); //Loop to continue search in the same List
 }while(choice == 2);  //Loop to continue search in another List of number
 cout<<"\n\tEND\n\n";
 return 0;
}
```

SAMPLE INPUT-OUTPUT



```
        LINEAR SEARCH

Enter the Number of Elements  : 10
Enter the elements one by one : 23 45 67 12 3 56 55 97 100 11

Enter the Number to search    : 66

-----------------------------------------------------------------------
The search element not found in the list
-----------------------------------------------------------------------

Do you continue search with same list - 1 ,another list - 2 or Exit - 3
Make your choice : 1


Enter the Number to search    : 67

-----------------------------------------------------------------------
The search element is found at 3 position
-----------------------------------------------------------------------

Do you continue search with same list - 1 ,another list - 2 or Exit - 3
Make your choice : 2


Enter the Number of Elements  : 5
Enter the elements one by one : 23 45 10 66 3

Enter the Number to search    : 5

-----------------------------------------------------------------------
The search element not found in the list
-----------------------------------------------------------------------

Do you continue search with same list - 1 ,another list - 2 or Exit - 3
Make your choice : 1


Enter the Number to search    : 66

-----------------------------------------------------------------------
The search element is found at 4 position
-----------------------------------------------------------------------

Do you continue search with same list - 1 ,another list - 2 or Exit - 3
Make your choice : 3


        END
```

BINARY SEARCH

AIM

To under the searching of a number using the Binary search technique. And to identify the comparison of item with the middle element of the main array and sub-array formed.

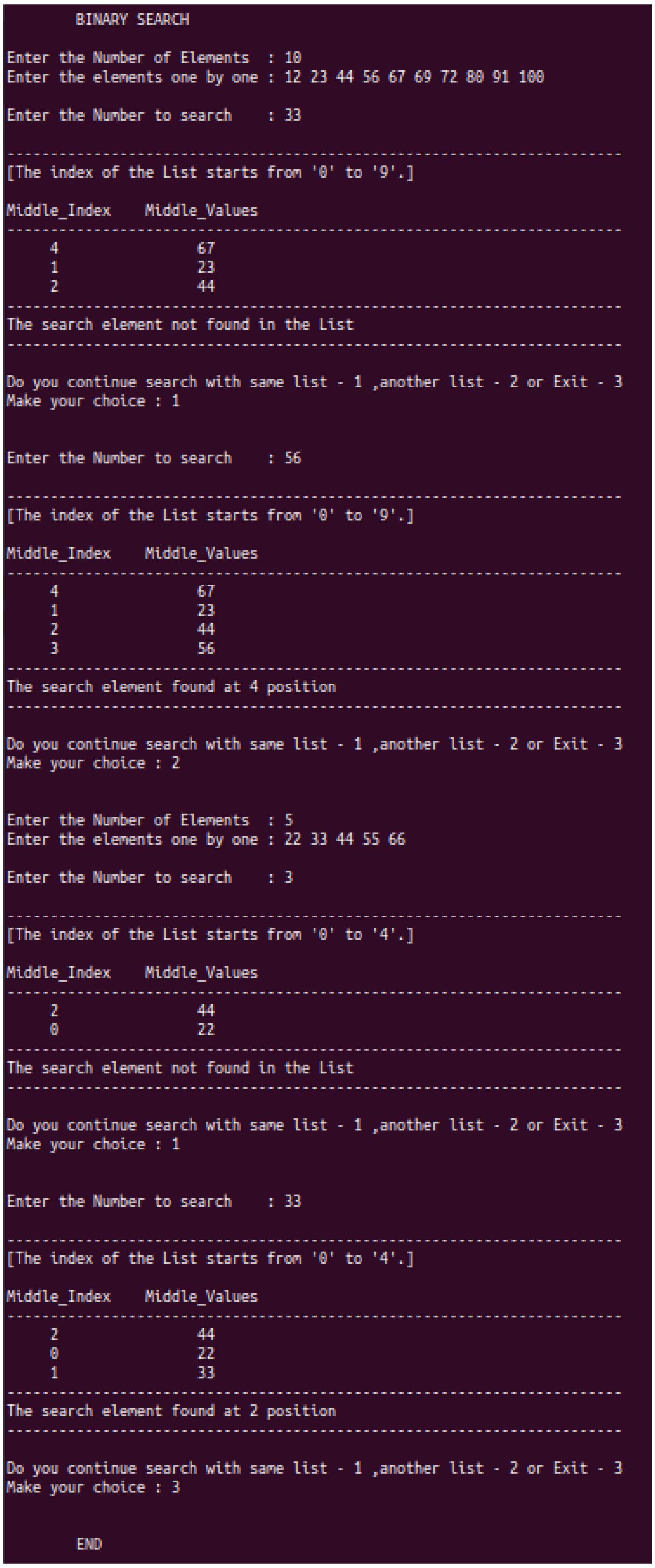
ALGORITHM

```
    BINARY_SEARCH( DATA, UB , ITEM)
        1.  set BEGIN = 0, END = UB
        2.  set MID = (BEGIN+END)/2
        3.  Repeat the steps   while DATA[MID]!=ITEM & BEGIN <= END
                (i)     if ITEM < DATA[MID]
                            (a) set END = MID-1
                (ii)    else
                            (a) set BEGIN = MID+1
                (iii)   end if condition
                (iv)    set MID = (BEGIN+END)/2
        4.  end while loop
        5.  if DATA[MID] = ITEM
                (i)     set POS = MID
                (ii)    print ITEM located at POS.
        6.  else
                (i)     set LOC = NULL
                (ii)    print ITEM not located.
        7.  end if condition
        8.  Exit
```

PROGRAM

```
#include<iostream>
#include<iomanip>
using namespace std;

void BINARY_SEARCH(int tempList[] , int upperLimit, int Key)
{
 int begin=0,end=upperLimit;

 cout<<"\n[The index of the List starts from '0' to '"<<upperLimit<<"'.]\n";
 cout<<"\nMiddle_Index"<<setw(18)<<"Middle_Values\n";

 cout<<"-----------------------------------------------"//
               "------------------------\n";
 int middle = ((begin+end)/2);

 while (tempList[middle] != Key && begin <= end)
 {
  cout<<setw(6)<<middle<<setw(18)<<tempList[middle]<<endl;

  if( Key < tempList[middle])
    end = middle - 1;
  else
   begin = middle + 1;

  middle = (begin+end)/2;
 }

 if(tempList[middle] == Key)
 {
   cout<<setw(6)<<middle<<setw(18)<<tempList[middle]<<endl;

   cout<<"---------------------------------------------"//
         "--------------------------\n";
   cout<<"The search element found at "<<middle+1<<" position ";
 }
 else
  {
   cout<<"------------------------------------------"//
        "-----------------------------\n";
   cout<<"The search element not found in the List";}
 }

int main()
{

 cout<<"\n\tBINARY SEARCH\n";

 int Number_of_terms,search_element,choice = 1,option=6, position;
 do{

 cout<<"\nEnter the Number of Elements  : ";
 cin>>Number_of_terms;

 int List[Number_of_terms];

 cout<<"Enter the elements one by one : ";

 for (int loop_variable = 0;loop_variable <
 Number_of_terms;loop_variable++)
 {
  cin>>List[loop_variable];
 }

 do
 {
 cout<<"\nEnter the Number to search    : ";
 cin>>search_element;
 cout<<"\n--------------------------------------------"//
        "--------------------------";

 BINARY_SEARCH(List,(Number_of_terms-1),search_element);

 cout<<"\n--------------------------------------------"//
        "--------------------------";
 cout<<endl;

 cout<<"\nDo you continue search with same list - 1 ,another list - 2"//
        " or Exit - 3 \nMake your choice : ";
 cin>>choice;
 cout<<"\n";

 }while(choice == 1 );

 }while(choice == 2);

 cout<<"\n\tEND\n\n";
 return 0;
}
```

SAMPLE INPUT-OUTPUT



```
        BINARY SEARCH

Enter the Number of Elements  : 10
Enter the elements one by one : 12 23 44 56 67 69 72 80 91 100

Enter the Number to search    : 33

-----------------------------------------------------------------------
[The index of the List starts from '0' to '9'.]

Middle_Index    Middle_Values
-----------------------------------------------------------------------
     4               67
     1               23
     2               44
-----------------------------------------------------------------------
The search element not found in the List
-----------------------------------------------------------------------

Do you continue search with same list - 1 ,another list - 2 or Exit - 3
Make your choice : 1


Enter the Number to search    : 56

-----------------------------------------------------------------------
[The index of the List starts from '0' to '9'.]

Middle_Index    Middle_Values
-----------------------------------------------------------------------
     4               67
     1               23
     2               44
     3               56
-----------------------------------------------------------------------
The search element found at 4 position
-----------------------------------------------------------------------

Do you continue search with same list - 1 ,another list - 2 or Exit - 3
Make your choice : 2


Enter the Number of Elements  : 5
Enter the elements one by one : 22 33 44 55 66

Enter the Number to search    : 3

-----------------------------------------------------------------------
[The index of the List starts from '0' to '4'.]

Middle_Index    Middle_Values
-----------------------------------------------------------------------
     2               44
     0               22
-----------------------------------------------------------------------
The search element not found in the List
-----------------------------------------------------------------------

Do you continue search with same list - 1 ,another list - 2 or Exit - 3
Make your choice : 1


Enter the Number to search    : 33

-----------------------------------------------------------------------
[The index of the List starts from '0' to '4'.]

Middle_Index    Middle_Values
-----------------------------------------------------------------------
     2               44
     0               22
     1               33
-----------------------------------------------------------------------
The search element found at 2 position
-----------------------------------------------------------------------

Do you continue search with same list - 1 ,another list - 2 or Exit - 3
Make your choice : 3


        END
```

TERNARY SEARCH

AIM

To under the searching of a number using the Ternary search technique. To evaluate the comparison of middle elements with the search key. Here the two middle element divides the array to three sub arrays.

ALGORITHM

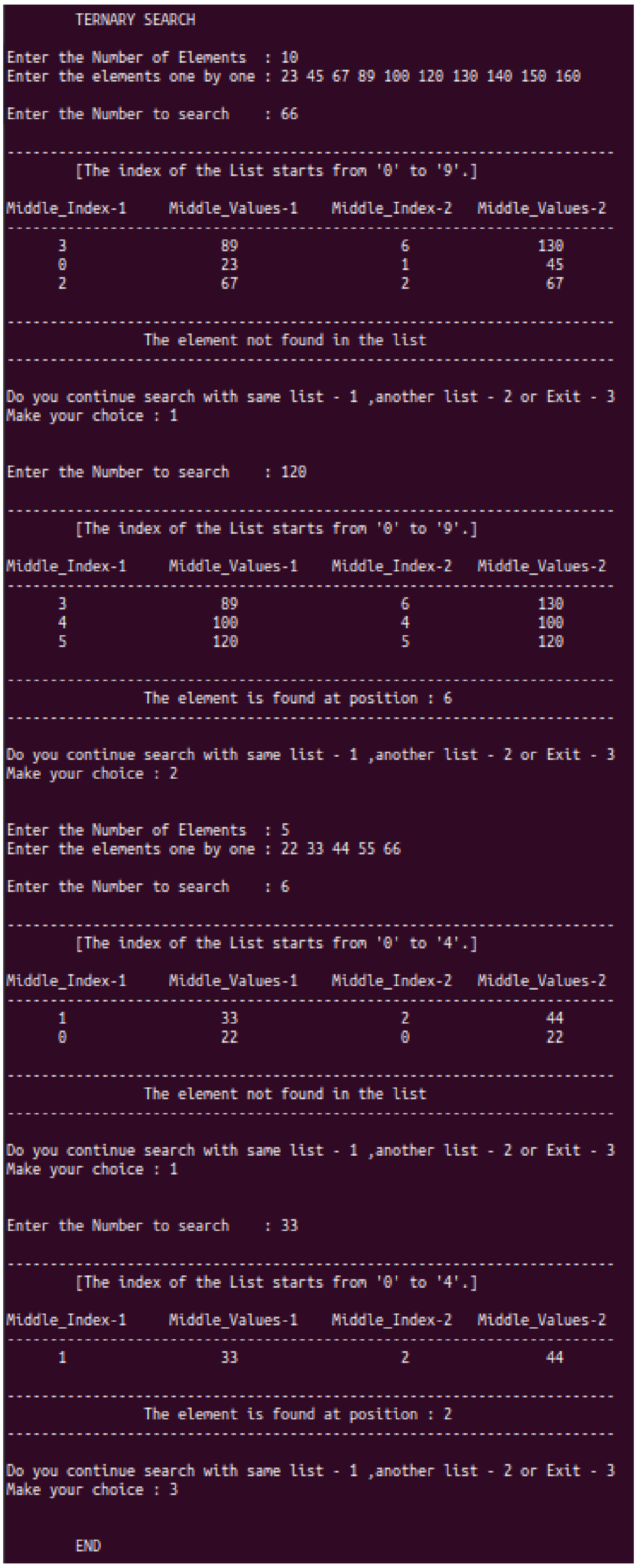
```
    TERNARY_SEARCH(DATA , ITEM , N )
        1.  set BEGIN = 1, END = N
        2.  Repeat the steps (i) to (x)   while BEGIN <= END
                (i)     set MID1 = ((END-BEGIN)/3)+BEGIN
                (ii)    set MID2 = ((2*(END - BEGIN))/3)=BEGIN
                (iii)   if ITEM = DATA[MID1]
                            (a) LOC = MID1
                            (b) return LOC
                (iv)    end if
                (v)     if ITEM = DATA[MID2]
                            (a) LOC = MID2
                            (b) return LOC
                (vi)    end if
                (vii)   if ITEM < DATA[MID1]
                            (a) END = MID1 - 1
                (viii)  else if ITEM > DATA[MID2]
                            (a) BEGIN = MID2 - 1
                (ix)    else
                            (a) BEGIN = MID1+1
                            (b) END = MID2-1
                (x)     end if condition
        3. end while loop
        4. set LOC=NULL
        5. return LOC
        6. Exit
```

PROGRAM

```
#include<iostream>
#include<iomanip>
using namespace std;

int TERTIARY_SEARCH(int tempList[] , int numberOfTerms , int searchKey)
```

```
{
 int begin=0 , end=numberOfTerms, middle1,middle2 ;

 cout<<"\n\t[The index of the List starts from '0' to '"<<//
           numberOfTerms<<"'.]\n";
 cout<<"\nMiddle_Index-1"<<setw(20)<<"Middle_Values-1"<<setw(18)<< //
           "Middle_Index-2"<<setw(18)<<"Middle_Values-2";
 cout<<"\n----------------------------------------------------" //
        "------------------\n";

 while(begin<=end)
 {
  middle1 = ((end - begin)/3)+begin;
  middle2 = ((2*(end-begin))/3)+begin;

  cout<<setw(7)<<middle1<<setw(20)<<tempList[middle1]<<setw(20) //
          <<middle2<<setw(18)<<tempList[middle2]<<endl;

  if(searchKey == tempList[middle1])
     return (middle1+1);
  if (searchKey == tempList[middle2])
      return (middle2+1);
  if (searchKey < tempList[middle1])
     {
      end = middle1-1;
     }
  else if (searchKey > tempList[middle2])
      {
       begin = middle2+1;
      }
  else
     {
      begin = middle1 +1;
      end = middle2 - 1;
     }
 }
 return -1;
}
 int main()
{
 cout<<"\n\tTERNARY SEARCH\n";
 int Number_of_terms,search_element,choice = 1,option=6, position;

 do{
 cout<<"\nEnter the Number of Elements  : ";
 cin>>Number_of_terms;
 int List[Number_of_terms];



 cout<<"Enter the elements one by one : ";
 for (int loop_variable = 0;loop_variable < Number_of_terms;loop_variable++)
 {
  cin>>List[loop_variable];
 }

 do {
 cout<<"\nEnter the Number to search    : ";
 cin>>search_element;

 cout<<"\n---------------------------------------------------"//
               "-------------------";
 position = TERTIARY_SEARCH( List , (Number_of_terms-1), search_element);
  cout<<"\n---------------------------------------------------"//
            "-------------------\n";
          if (position != -1)
    cout<<"\t\tThe element is found at position : "<<position;
 else
    cout<<"\t\tThe element not found in the list ";
 cout<<"\n---------------------------------------------------"//
          "-------------------";
 cout<<endl;

 cout<<"\nDo you continue search with same list - 1 ,another list - 2"//
         " or Exit - 3 \nMake your choice : ";
 cin>>choice;
 cout<<"\n";

 }while(choice == 1 );
 }while(choice == 2);
 cout<<"\n\tEND\n\n";
 return 0;
}
```

SAMPLE INPUT-OUTPUT



```
        TERNARY SEARCH

Enter the Number of Elements  : 10
Enter the elements one by one : 23 45 67 89 100 120 130 140 150 160

Enter the Number to search    : 66

-----------------------------------------------------------------------
        [The index of the List starts from '0' to '9'.]

Middle_Index-1     Middle_Values-1    Middle_Index-2   Middle_Values-2
-----------------------------------------------------------------------
      3                 89                  6               130
      0                 23                  1                45
      2                 67                  2                67

-----------------------------------------------------------------------
                The element not found in the list
-----------------------------------------------------------------------

Do you continue search with same list - 1 ,another list - 2 or Exit - 3
Make your choice : 1


Enter the Number to search    : 120

-----------------------------------------------------------------------
        [The index of the List starts from '0' to '9'.]

Middle_Index-1     Middle_Values-1    Middle_Index-2   Middle_Values-2
-----------------------------------------------------------------------
      3                 89                  6               130
      4                100                  4               100
      5                120                  5               120

-----------------------------------------------------------------------
                The element is found at position : 6
-----------------------------------------------------------------------

Do you continue search with same list - 1 ,another list - 2 or Exit - 3
Make your choice : 2


Enter the Number of Elements  : 5
Enter the elements one by one : 22 33 44 55 66

Enter the Number to search    : 6

-----------------------------------------------------------------------
        [The index of the List starts from '0' to '4'.]

Middle_Index-1     Middle_Values-1    Middle_Index-2   Middle_Values-2
-----------------------------------------------------------------------
      1                 33                  2                44
      0                 22                  0                22

-----------------------------------------------------------------------
                The element not found in the list
-----------------------------------------------------------------------

Do you continue search with same list - 1 ,another list - 2 or Exit - 3
Make your choice : 1


Enter the Number to search    : 33

-----------------------------------------------------------------------
        [The index of the List starts from '0' to '4'.]

Middle_Index-1     Middle_Values-1    Middle_Index-2   Middle_Values-2
-----------------------------------------------------------------------
      1                 33                  2                44

-----------------------------------------------------------------------
                The element is found at position : 2
-----------------------------------------------------------------------

Do you continue search with same list - 1 ,another list - 2 or Exit - 3
Make your choice : 3


        END
```

INTERPOLATION SEARCH

AIM

To under stand the searching a number using the Interpolation Search algorithm. Also to evaluate the approximate position determined by the interpolation search equation.

ALGORITHM

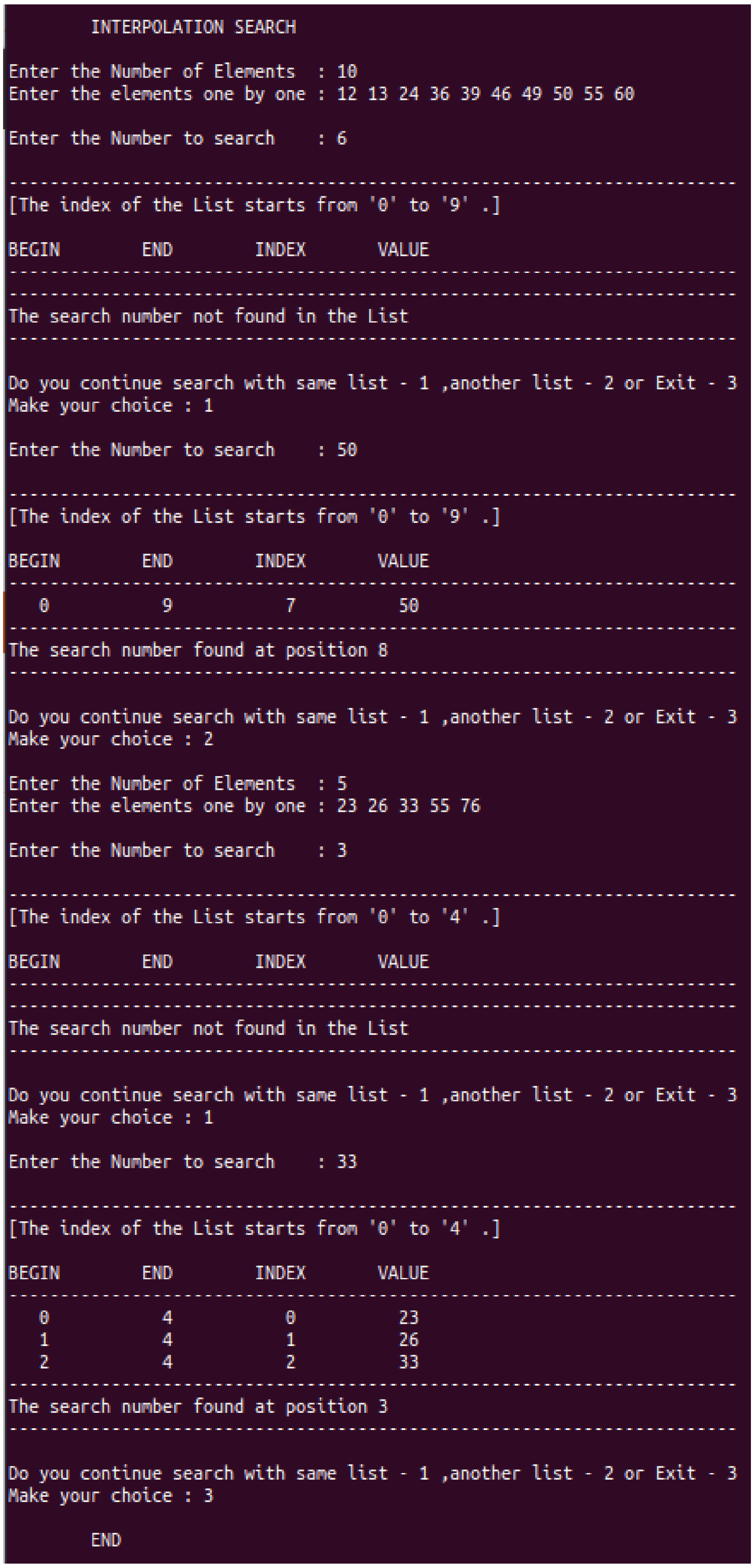
```
    INTERPOLATION_SEARCH(DATA, N, ITEM)
        1.  set BEGIN = 1 , END = N
        2. Repeat the steps  while BEGIN <= END and ITEM >= DATA[BEGIN]
        and ITEM <= DATA[END]
            (i)     set POS = BEGIN + (((ITEM - DATA[BEGIN])/
            (DATA[END] - DATA[BEGIN]))*(END - BEGIN))
            (ii)   if ITEM=DATA[POS]
                        (a) set LOC = POS
                        (b) return LOC
            (iii)  else if ITEM > DATA[POS]
                        (a) BEGIN = POS+1
            (iv)   else
                        (a) END = POS - 1
            (v)    end if condition
        3.  end while loop
        4.  set LOC = NULL
        5.  return LOC
        6.  Exit
```

PROGRAM

```
#include<iostream>
#include<iomanip>
using namespace std;

int INTERPOLATION_SEARCH(int tempList[] , int tempTerms, int search)
{

 cout<<"\n[The index of the List starts from '0' to '"//
           <<tempTerms-1<<"' .]\n";
 int end = tempTerms-1 , begin = 0;

 cout<<"\nBEGIN"<<setw(11)<<"END"<<setw(13)<<"INDEX"//
```

```
           <<setw(14)<<"VALUE \n";
 cout<<"------------------------------------------------------"//
           "-----------------\n";

 while(begin <= end & search >= tempList[begin] & search <= tempList[end])
 {

  float valuenum = search - tempList[begin];
  float valueden = tempList[end]-tempList[begin];
  float value1 = end-begin;
  float value2 = ((valuenum)/(valueden)) * value1;
  int position = begin+value2;

  cout<<setw(4)<<begin<<setw(12)<<end<<setw(12)<<//
           position<<setw(12)<<tempList[position]<<endl;

  if (search == tempList[position])
  {
    cout<<"--------------------------------------------------"//
               "---------------------\n";
   return position+1;
  }
  else if (search > tempList[position])
  {
   begin = position+1;
  }
  else
  {
   end = position - 1;
  }
 }
 cout<<"------------------------------------------------------"//
           "-----------------\n";
 return -1;
}

int main()
{
 cout<<"\n\tINTERPOLATION SEARCH\n";

 int Number_of_terms,search_element,choice = 1,option=6, position;
 do
 {
 cout<<"\nEnter the Number of Elements  : ";
 cin>>Number_of_terms;

 int List[Number_of_terms];

 cout<<"Enter the elements one by one : ";

 for (int loop_variable = 0;loop_variable < Number_of_terms;loop_variable++)
 {
  cin>>List[loop_variable];
 }

 do
 {
 cout<<"\nEnter the Number to search    : ";
 cin>>search_element;

 cout<<"\n----------------------------------------------------"//
        "--------------------";

 position = INTERPOLATION_SEARCH(List, Number_of_terms, search_element);
       if(position != -1)
    cout<<"The search number found at position "<<position;
      else
   cout<<"The search number not found in the List ";
 cout<<"\n----------------------------------------------------"//
            "--------------------";
 cout<<endl;
 cout<<"\nDo you continue search with same list - 1 ,another list - 2 "//
            "or Exit - 3 \nMake your choice : ";
 cin>>choice;

 }while(choice == 1 );
 }while(choice == 2);
 cout<<"\n\tEND\n\n";
 return 0;
}
```

SAMPLE INPUT-OUTPUT



```
        INTERPOLATION SEARCH

Enter the Number of Elements  : 10
Enter the elements one by one : 12 13 24 36 39 46 49 50 55 60

Enter the Number to search    : 6

----------------------------------------------------------------------
[The index of the List starts from '0' to '9' .]

BEGIN        END        INDEX       VALUE
----------------------------------------------------------------------
----------------------------------------------------------------------
The search number not found in the List
----------------------------------------------------------------------

Do you continue search with same list - 1 ,another list - 2 or Exit - 3
Make your choice : 1

Enter the Number to search    : 50

----------------------------------------------------------------------
[The index of the List starts from '0' to '9' .]

BEGIN        END        INDEX       VALUE
----------------------------------------------------------------------
   0           9           7          50
----------------------------------------------------------------------
The search number found at position 8
----------------------------------------------------------------------

Do you continue search with same list - 1 ,another list - 2 or Exit - 3
Make your choice : 2

Enter the Number of Elements  : 5
Enter the elements one by one : 23 26 33 55 76

Enter the Number to search    : 3

----------------------------------------------------------------------
[The index of the List starts from '0' to '4' .]

BEGIN        END        INDEX       VALUE
----------------------------------------------------------------------
----------------------------------------------------------------------
The search number not found in the List
----------------------------------------------------------------------

Do you continue search with same list - 1 ,another list - 2 or Exit - 3
Make your choice : 1

Enter the Number to search    : 33

----------------------------------------------------------------------
[The index of the List starts from '0' to '4' .]

BEGIN        END        INDEX       VALUE
----------------------------------------------------------------------
   0           4           0          23
   1           4           1          26
   2           4           2          33
----------------------------------------------------------------------
The search number found at position 3
----------------------------------------------------------------------

Do you continue search with same list - 1 ,another list - 2 or Exit - 3
Make your choice : 3

        END
```

FIBONACCI SEARCH

AIM

To understand the searching of numbers using the Fibonacci Search Algorithm. Also to estimate how the Fibonacci number variation effect with the speed of searching technique.

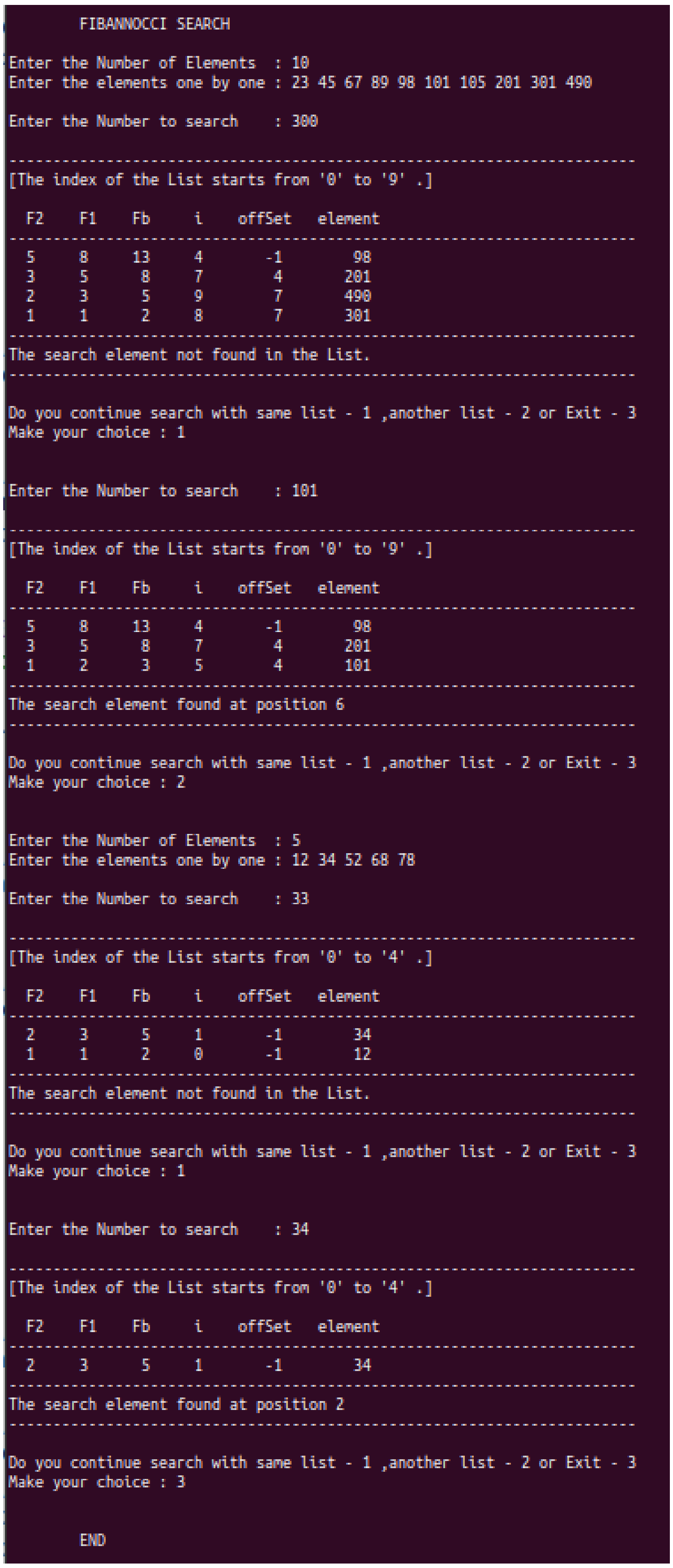
ALGORITHM

```
FIBONACCI_SEARCH(DATA,ITEM,N,LOC)
    1.  Set f1 = 1
    2.  Set f2 = 0
    3.  Set f = f1+f2
    4.  Set an offset = -1
    5.  Repeat the steps while f < N then
            (i)     Set f2 = f1
            (ii)    Set f1 = f
            (iii)   Set f = f1+f2
    6.  End while loop
    7.  Repeat the step while f >1 then
            (i)     Set i = offset + f2
            (ii)    if DATA[i] < ITEM then
                        (a) Set f = f1
                        (b) Set f1 = f2
                        (c) Set f2 = f - f1
                        (d) Set offset = i
            (iii)   else if DATA[i] > ITEM then
                        (a) Set f = f2
                        (b) Set f1 = f - f2
                        (c) Set f2 = f - f1
            (iv)    else
                        (a) Set LOC = i
                        (b) Set RETURN LOC
            (v)     End if condition
    8.  End while loop
    9.  if f1 = 1 and DATA[offset + 1] = ITEM then
            (i)     Set LOC = offset + 1
            (ii)    RETURN LOC
    10. End if condition
    11. RETURN -1
    12. Exit
```

PROGRAM

```
#include<iostream>

#include<iomanip>

using namespace std;

int FIBONACCI_SEARCH(int tempList[] , int tempTerms ,  int Key)
{
 cout<<"\n[The index of the List starts from '0' to '"//
            <<tempTerms-1<<"' .]\n";
 int firstElement = 0 , secondElement = 1;
 int thirdElement = firstElement+secondElement;
 int offSet = -1 , i;

 while (thirdElement < tempTerms)
 {
  firstElement = secondElement;
  secondElement = thirdElement;
  thirdElement = firstElement+secondElement;
 }

 cout<<"\n  F2"<<setw(6)<<"F1"<<setw(6)<<"Fb"<<setw(6)<<"i"//
               <<setw(10)<<"offSet"<<setw(12)<<"element \n";
 cout<<"---------------------------------------------------"//
               "--------------------\n";

 while(thirdElement > 1)
 {

  i = offSet + firstElement;
  cout<<"  "<<firstElement<<setw(6)<<secondElement<<setw(7)//
       << thirdElement<<setw(6)<<i<<setw(9)<<offSet<<setw(10)<<tempList[i]<<"\n";

  if (tempList[i] < Key)
  {
   thirdElement = secondElement;
   secondElement = firstElement;
   firstElement = thirdElement - secondElement;
   offSet = i;
  }

  else if (tempList[i] > Key)
  {
   thirdElement = firstElement;
   secondElement = secondElement - firstElement;
   firstElement = thirdElement - secondElement;
  }

  else
  {  cout<<"-----------------------------------------------"//
            "------------------------\n";
   return i+1;
  }

 }
 if (secondElement == 1 & (tempList[offSet+2]) == Key)
 { cout<<"--------------------------------------------------"//
            "---------------------\n";
  return (offSet+2);
 }
 cout<<"-----------------------------------------------------"//
            "------------------\n";
 return (-1);
}

int main()
{
 cout<<"\n\tFIBANNOCCI SEARCH\n";
 int Number_of_terms,search_element,choice = 1,option=6, position;

 do
 {
 cout<<"\nEnter the Number of Elements  : ";
 cin>>Number_of_terms;

 int List[Number_of_terms];

 cout<<"Enter the elements one by one : ";

 for (int loop_variable = 0;loop_variable < Number_of_terms;loop_variable++)
 {
  cin>>List[loop_variable];
 }

 do
 {
 cout<<"\nEnter the Number to search    : ";
 cin>>search_element;


 cout<<"\n--------------------------------------------------"//
               "-------------------";
  position = FIBONACCI_SEARCH(List , Number_of_terms, search_element );
           if (position != -1)
  {
   cout<<"The search element found at position "<<position;
 }
    else
 {
   cout<<"The search element not found in the List. ";
 }

 cout<<"\n-----------------------------------------------------"//
               "----------------";
 cout<<endl;

 cout<<"\nDo you continue search with same list - 1 ,another list - 2 "//
               "or Exit - 3 \nMake your choice : ";
 cin>>choice;
 cout<<"\n";

 }while(choice == 1 );

 }while(choice == 2);

 cout<<"\n\tEND\n\n";

 return 0;

}
```

SAMPLE INPUT-OUTPUT



```
        FIBANNOCCI SEARCH

Enter the Number of Elements  : 10
Enter the elements one by one : 23 45 67 89 98 101 105 201 301 490

Enter the Number to search    : 300

-----------------------------------------------------------------------
[The index of the List starts from '0' to '9' .]

  F2    F1    Fb     i    offSet   element
-----------------------------------------------------------------------
  5     8     13     4       -1        98
  3     5      8     7        4       201
  2     3      5     9        7       490
  1     1      2     8        7       301
-----------------------------------------------------------------------
The search element not found in the List.
-----------------------------------------------------------------------

Do you continue search with same list - 1 ,another list - 2 or Exit - 3
Make your choice : 1


Enter the Number to search    : 101

-----------------------------------------------------------------------
[The index of the List starts from '0' to '9' .]

  F2    F1    Fb     i    offSet   element
-----------------------------------------------------------------------
  5     8     13     4       -1        98
  3     5      8     7        4       201
  1     2      3     5        4       101
-----------------------------------------------------------------------
The search element found at position 6
-----------------------------------------------------------------------

Do you continue search with same list - 1 ,another list - 2 or Exit - 3
Make your choice : 2


Enter the Number of Elements  : 5
Enter the elements one by one : 12 34 52 68 78

Enter the Number to search    : 33

-----------------------------------------------------------------------
[The index of the List starts from '0' to '4' .]

  F2    F1    Fb     i    offSet   element
-----------------------------------------------------------------------
  2     3      5     1       -1        34
  1     1      2     0       -1        12
-----------------------------------------------------------------------
The search element not found in the List.
-----------------------------------------------------------------------

Do you continue search with same list - 1 ,another list - 2 or Exit - 3
Make your choice : 1


Enter the Number to search    : 34

-----------------------------------------------------------------------
[The index of the List starts from '0' to '4' .]

  F2    F1    Fb     i    offSet   element
-----------------------------------------------------------------------
  2     3      5     1       -1        34
-----------------------------------------------------------------------
The search element found at position 2
-----------------------------------------------------------------------

Do you continue search with same list - 1 ,another list - 2 or Exit - 3
Make your choice : 3


        END
```

BUBBLE SORT

AIM

To understand the sorting of the numbers using Bubble Sort algorithm.

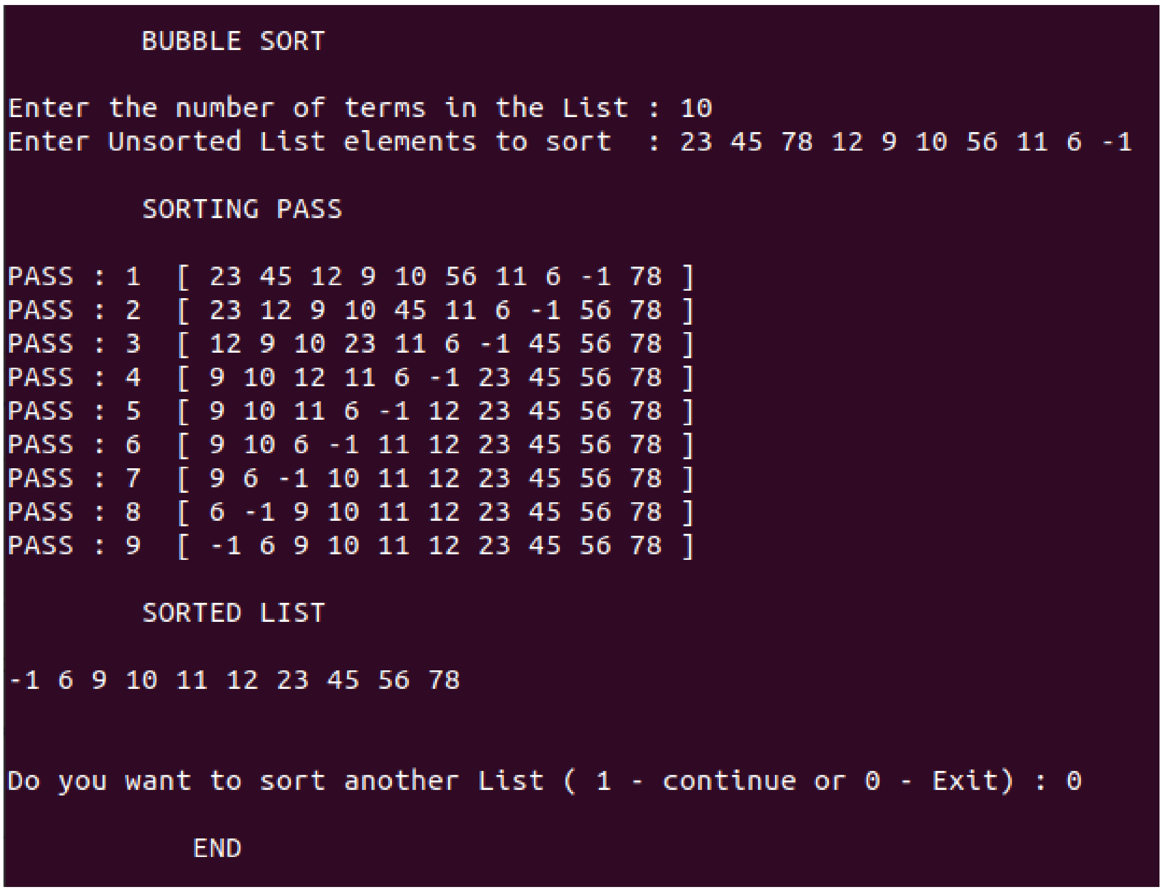
ALGORITHM

```
BUBBLE_SORT(DATA, N)
    1.  set J = 1, I = 1
    2.  Repeat the steps (i) and (ii) for I to N
            (i)     Repeat the steps (a) and (b) for J to N
                    (a) if DATA[J] > DATA[J+1]
                        * set TEMP = DATA[J+1]
                        * set DATA[J+1] = DATA[J]
                        * set DATA[J] = TEMP
                    (b) end if condition
            (ii)    end for loop
    3.  end for loop
    4.  Exit
```

PROGRAM

```
#include <iostream>
using namespace std;

void BUBBLE_SORT(int tempList[], int upperLimit)
{
 cout<<"\n\tSORTING PASS\n\n";
 for (int i = 0 ; i < upperLimit ; i++)
 {
  for (int j = 0 ; j < upperLimit - i ; j++)
  {
   if (tempList[j] > tempList[j+1])
   {
    int temp = tempList[j+1];
    tempList[j+1] = tempList[j];
    tempList[j] = temp;
   }
  }
  cout<<"PASS : "<<i+1<<"  [ ";
  for(int index = 0;index<upperLimit+1;index++)
 {
  cout<<tempList[index]<<" ";
 }
 cout<<"]\n";
 }
}

int main()
{
 cout<<"\n\tBUBBLE SORT\n";
 int terms , index , choice , option;
  do
  {
   cout<<"\nEnter the number of terms in the List : ";
   cin>>terms;
   int List[terms];
   int B[terms];
   cout<<"Enter Unsorted List elements to sort  : ";

   for(index = 0;index<terms;index++)
   {
     cin>>List[index];
   }

   BUBBLE_SORT(List, terms-1);

   cout<<"\n\tSORTED LIST\n\n";
       for ( index = 0;index < terms ; index++)
       {
          cout<<List[index]<<" ";
       }
       cout<<endl;
   cout<<"\n\nDo you want to sort another List"//
             "( 1 - continue or 0 - Exit) : ";
   cin>>option;

  }while(option!=0);
 cout<<"\n\t   END\n\n";
 return 0;
}
```

SAMPLE INPUT-OUTPUT



```
        BUBBLE SORT

Enter the number of terms in the List : 10
Enter Unsorted List elements to sort  : 23 45 78 12 9 10 56 11 6 -1

        SORTING PASS

PASS : 1  [ 23 45 12 9 10 56 11 6 -1 78 ]
PASS : 2  [ 23 12 9 10 45 11 6 -1 56 78 ]
PASS : 3  [ 12 9 10 23 11 6 -1 45 56 78 ]
PASS : 4  [ 9 10 12 11 6 -1 23 45 56 78 ]
PASS : 5  [ 9 10 11 6 -1 12 23 45 56 78 ]
PASS : 6  [ 9 10 6 -1 11 12 23 45 56 78 ]
PASS : 7  [ 9 6 -1 10 11 12 23 45 56 78 ]
PASS : 8  [ 6 -1 9 10 11 12 23 45 56 78 ]
PASS : 9  [ -1 6 9 10 11 12 23 45 56 78 ]

        SORTED LIST

-1 6 9 10 11 12 23 45 56 78


Do you want to sort another List ( 1 - continue or 0 - Exit) : 0

           END
```

SELECTION SORT

AIM

To understand the sorting of elements using the Selection sort.

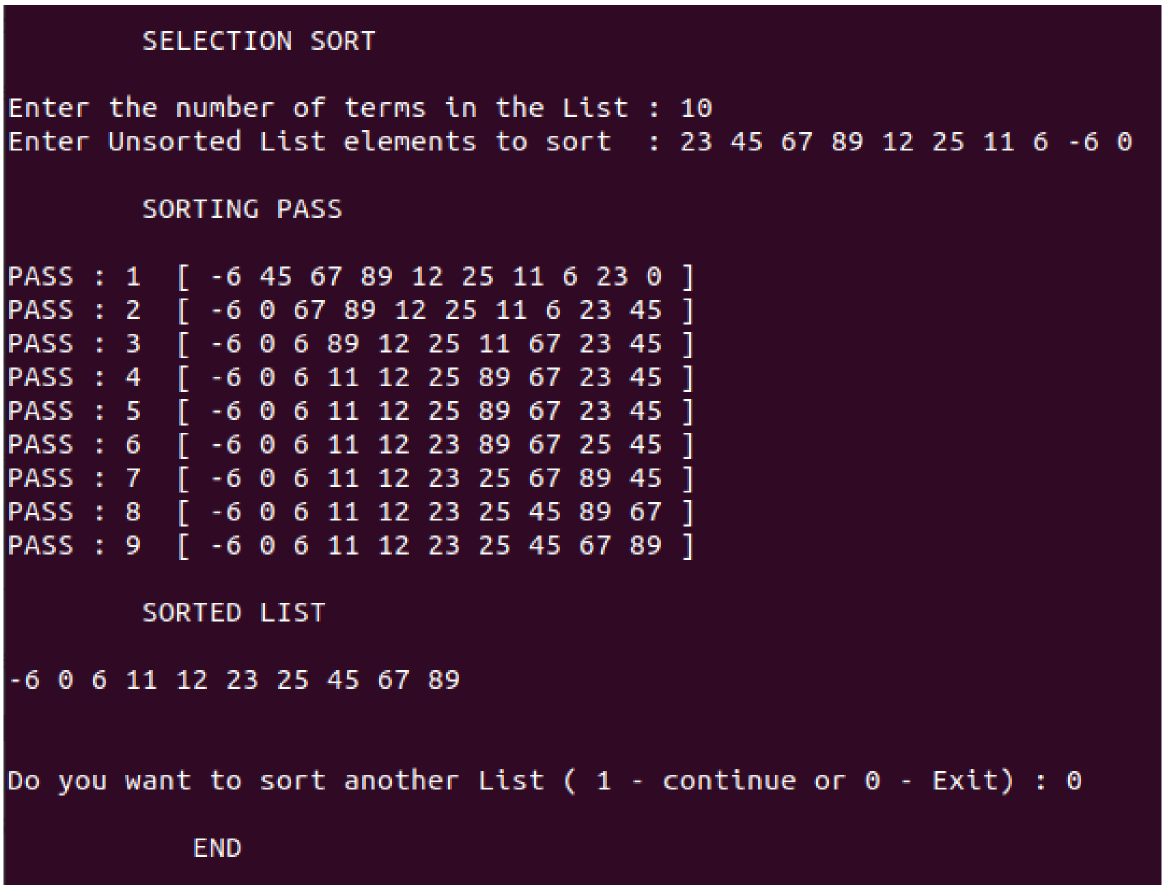
ALGORITHM

```
    SELECTION_SORT(DATA, N)
        1.  set I = 1, J = I+1
        2.  Repeat the steps (i) to (vi) for I to N-1
                (i)     set MIN = I
                (ii)    Repeat the steps (a) and (b) for J to N
                            (a) if DATA[J]  = DATA[MIN]
                                    * set MIN = J
                            (b) end if condition
                (iii)   end for loop
                (iv)    set TEMP = DATA[I]
                (v)     set DATA[I] = DATA[MIN]
                (vi)    set DATA[MIN] = TEMP
        3.  end for loop
        4.  Exit
```

PROGRAM

```
#include <iostream>
using namespace std;

void SELECTION_SORT(int tempList[] , int upperLimit)
{
int temp,min;

 cout<<"\n\tSORTING PASS\n\n";

for (int i=0;i<upperLimit-1;i++)
{
 min = i ;
 for( int j=i+1;j<upperLimit;j++)
 {
  if (tempList[j]<tempList[min])
  {
   min = j;
  }
 }
 temp = tempList[i];
 tempList[i] = tempList[min];
 tempList[min] = temp;

 cout<<"PASS : "<<i+1<<"  [ ";
 for(int index =0;index<upperLimit;index++)
 {
  cout<<tempList[index] << " ";
 }
 cout<<"]"<<endl;
}
}

int main()
{
 cout<<"\n\tSELECTION SORT\n";
 int terms , index , choice , option;
 do
 {
   cout<<"\nEnter the number of terms in the List : ";
   cin>>terms;

   int List[terms];
   int B[terms];

   cout<<"Enter Unsorted List elements to sort  : ";

   for(index = 0;index<terms;index++)
   {
     cin>>List[index];
   }

        SELECTION_SORT(List,terms);

       cout<<"\n\tSORTED LIST\n\n";
       for ( index = 0;index < terms ; index++)
       {
          cout<<List[index]<<" ";
       }
       cout<<endl;
   cout<<"\n\nDo you want to sort another "//
               "List ( 1 - continue or 0 - Exit) : ";
   cin>>option;

  }while(option!=0);
 cout<<"\n\t   END\n\n";
 return 0;
}
```

SAMPLE INPUT-OUTPUT



```
        SELECTION SORT

Enter the number of terms in the List : 10
Enter Unsorted List elements to sort  : 23 45 67 89 12 25 11 6 -6 0

        SORTING PASS

PASS : 1  [ -6 45 67 89 12 25 11 6 23 0 ]
PASS : 2  [ -6 0 67 89 12 25 11 6 23 45 ]
PASS : 3  [ -6 0 6 89 12 25 11 67 23 45 ]
PASS : 4  [ -6 0 6 11 12 25 89 67 23 45 ]
PASS : 5  [ -6 0 6 11 12 25 89 67 23 45 ]
PASS : 6  [ -6 0 6 11 12 23 89 67 25 45 ]
PASS : 7  [ -6 0 6 11 12 23 25 67 89 45 ]
PASS : 8  [ -6 0 6 11 12 23 25 45 89 67 ]
PASS : 9  [ -6 0 6 11 12 23 25 45 67 89 ]

        SORTED LIST

-6 0 6 11 12 23 25 45 67 89


Do you want to sort another List ( 1 - continue or 0 - Exit) : 0

           END
```

INSERTION SORT

AIM

To understand the sorting algorithm for number using Insertion sort.

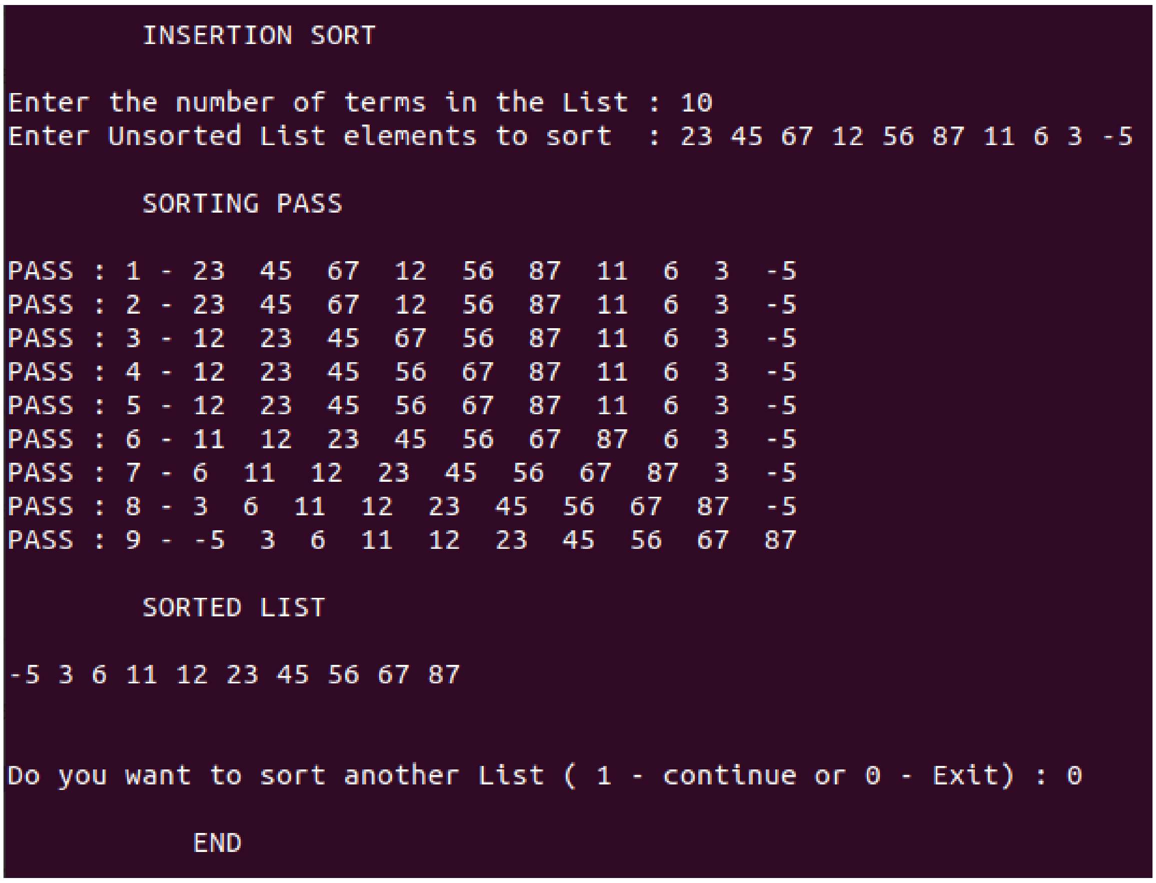
ALGORITHM

```
    INSERTION_SORT(DATA, N)
        1.  set K = 1
        2.  Repeat the steps (i) to (iv) while K != N
                (i)     set TEMP = K
                (ii)    Repeat the following steps (a) to (d)
                          while DATA[K] < DATA[K-1] and K != 0
                        (a)   set TEMP = DATA[K-1]
                        (b)   set DATA[K-1] = DATA[K]
                        (c)   set DATA[k] = TEMP
                        (d)   set K = K-1
                (iii)   end while loop
                (iv)    set K = TEMP
        3. end while loop
        4. Exit
```

PROGRAM

```
#include <iostream>
using namespace std;

void INSERTION_SORT(int tempList[] , int upperLimit)
{
  cout<<"\n\tSORTING PASS\n\n";
 int temp , TEMP;
 int    k = 1;
 while (k!= upperLimit)
 {
  temp = k;
  while( tempList[k] < tempList[k-1] & k != 0)
  {
   TEMP = tempList[k-1];
   tempList[k-1] = tempList[k];
   tempList[k] =  TEMP;
   k = k-1;
  }
  k = temp;
  cout<<"PASS : "<<k<<" - ";
  for(int element = 0 ; element < upperLimit ; element++)
  {
   cout<<tempList[element]<<"  ";
   temp = temp+1;
  }
  cout<<endl;
  k = k+1;
 }
}

int main()
{
 cout<<"\n\tINSERTION SORT\n";
 int terms , index , option;
 do
 {
   cout<<"\nEnter the number of terms in the List : ";
   cin>>terms;

   int List[terms];
   int B[terms];

   cout<<"Enter Unsorted List elements to sort  : ";

   for(index = 0;index<terms;index++)
   {
     cin>>List[index];
   }

         INSERTION_SORT(List , terms);



       cout<<"\n\tSORTED LIST\n\n";
       for ( index = 0;index < terms ; index++)
       {
          cout<<List[index]<<" ";
       }
       cout<<endl;
   cout<<"\n\nDo you want to sort another List "//
               "( 1 - continue or 0 - Exit) : ";
   cin>>option;

  }while(option!=0);
 cout<<"\n\t   END\n\n";
 return 0;
}
```

SAMPLE INPUT-OUTPUT



```
        INSERTION SORT

Enter the number of terms in the List : 10
Enter Unsorted List elements to sort  : 23 45 67 12 56 87 11 6 3 -5

        SORTING PASS

PASS : 1 - 23  45  67  12  56  87  11  6  3  -5
PASS : 2 - 23  45  67  12  56  87  11  6  3  -5
PASS : 3 - 12  23  45  67  56  87  11  6  3  -5
PASS : 4 - 12  23  45  56  67  87  11  6  3  -5
PASS : 5 - 12  23  45  56  67  87  11  6  3  -5
PASS : 6 - 11  12  23  45  56  67  87  6  3  -5
PASS : 7 - 6  11  12  23  45  56  67  87  3  -5
PASS : 8 - 3  6  11  12  23  45  56  67  87  -5
PASS : 9 - -5  3  6  11  12  23  45  56  67  87

        SORTED LIST

-5 3 6 11 12 23 45 56 67 87


Do you want to sort another List ( 1 - continue or 0 - Exit) : 0

           END
```

MERGE SORT

AIM

To understand the sorting of numbers using the MERGE SORT algorithm.

ALGORITHM

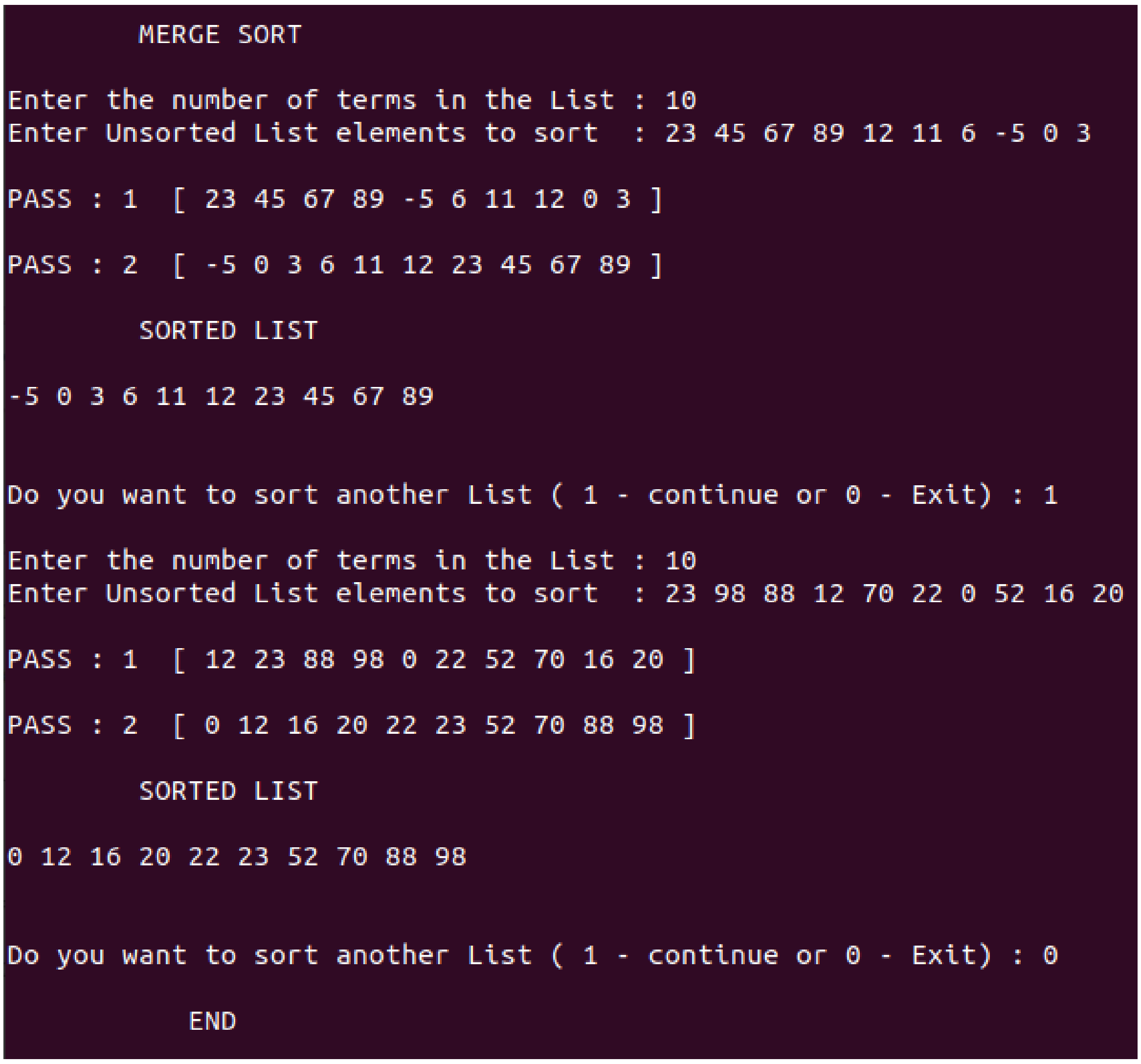
```
MERGE(DATA_A, N_A, LBA, DATA_B, N_B, LBB, DATA_C, LBC)
    1.  set NA = LBA
    2.  set NB = LBB
    3.  set PTR = LBC
    4.  set UBA = LBA + N_A - 1
    5.  set UBB = LBB + N_B - 1
    6.  Repeat the steps while NA <= UBA && NB <=UBB
            (i)     if DATA_A[NA] < DATA_B[NB]
                        (a) DATA_C[PTR]  =DATA_A[NA]
                        (b) NA = NA + 1
                        (c) PTR = PTR + 1
            (ii)    else
                        (a) DATA_C[PTR]  =DATA_B[AB]
                        (b) NB = NB + 1
                        (c) PTR = PTR + 1
            (iii)   end if condition
    7.  end while loop
    8.  if NA > UBA then
            (i)     set I = 0
            (ii)    Repeat the steps for I to UBA-NB
                        (a) DATA_C[PTR + I] = DATA_B[NB+I]
            (iii)   End for loop
    7.  else
            (i)     Repeat the steps for I=0 to UBA-NA
                        (a) DATA_C[PTR + I] = DATA_A[NA+I]
            (ii)    End for loop
    8.  End if condition
    9.  Exit


MERGE_PASS(A , N , L , B)
    1.  Set Q = floor(N/2*L)
    2.  S = 2*L*Q
    3.  R = N { S
```

```
        4. Repeat the following for i=1 to i<Q then
                (i)     set LB = 1 +((2*i) { 2)L
                (ii)    MERGE(A , L , LB , A , L , LB + L , B , LB)
        5.  End for loop
        6.  if R <= L then
                (i)     Repeat the steps for j=0 to j<R
                            (a) Set B[S+j] = A[S+j]
                (ii)    End the for loop
        7.  else
                (i) MERGING(A , L , S + 1 , A , R { L , L + S + 1, B , S + 1)
        8.  End if condition
        9.  Exit

    MERGE_SORT(A , N)
        1. set L = 1
        2. Repeat the steps while L < N
            (i)     MERGE_PASS(A , N , L , B)
            (ii)    MERGE_PASS(B , N , 2*L , A)
            (iii)   set L = 4*L
        3.  End while loop
        4.  Exit
```

PROGRAM

```
#include <iostream>
using namespace std;

void MERGE(int A[] , int Num_A , int LBA ,int B[] , int Num_B ,//
                int LBB , int *C , int LBC)
{

  int temp_A = LBA;
  int temp_B = LBB;
  int PTR = LBC;
  int UBA = LBA + Num_A - 1;
  int UBB = LBB + Num_B - 1;

  while (temp_A <= UBA && temp_B <= UBB)
  {
    if( A[temp_A] < B[temp_B])
    {
      C[PTR] = A[temp_A];
      temp_A =temp_A + 1;
      PTR = PTR+1;
    }
    else
    {
      C[PTR] = B[temp_B];
      temp_B = temp_B + 1;
      PTR = PTR +1;
    }
  }

  if (temp_A > UBA)
  {
    for (int i =0;i <= UBB - temp_B;i++)
    {
     C[PTR + i ] = B[temp_B + i];
    }
  }

  else
  {
    for(int i=0 ;i <= UBA - temp_A; i++)
    {
      C[PTR + i ] = A[temp_A + i];
    }
  }
}

void MERGEPASS( int *A , int Num_A , int Num_SubA , int *B)
{

  int Q = Num_A/(2*Num_SubA);
  int S = 2*Num_SubA*Q;
  int R = Num_A - S;
  int LB;

  for( int i=1;i<=Q;i++)
  {
    LB = (2*i-2)*Num_SubA;
    MERGE(A , Num_SubA , LB , A , Num_SubA , LB+Num_SubA , B , LB);
  }

  if( R <= Num_SubA)
  {
    for(int j =0 ; j< R ; j++)
     {
            B[S+j] = A[S+j];
     }
   }
  else
  {
    MERGE(A, Num_SubA , S ,A, R-Num_SubA, Num_SubA + S, B ,S);
  }
}

void MERGESORT(int *A, int *B , int NumA)
{
  int L=1;
  int j=0;

  while (L < NumA)
  {
    MERGEPASS( A , NumA , L , B);

    MERGEPASS(B , NumA , (2*L) , A);

    L = 4*L;

    cout<<"\nPASS : "<<j+1<<"  [ ";

    for(int i=0;i<NumA;i++)
    {
      cout<<A[i]<<" ";
    }
    j = j+1;
    cout<<"]\n";
  }
}

int main()
{
 cout<<"\n\tMERGE SORT\n";
 int terms , index , choice , option;
 do
 {
   cout<<"\nEnter the number of terms in the List : ";

   cin>>terms;

   int List[terms];

   int B[terms];

   cout<<"Enter Unsorted List elements to sort  : ";

   for(index = 0;index<terms;index++)
   {
     cin>>List[index];
   }

        MERGESORT( List , B , terms);

    cout<<"\n\tSORTED LIST\n\n";

    for ( index = 0;index < terms ; index++)
       {
          cout<<List[index]<<" ";
       }

    cout<<endl;
   cout<<"\n\nDo you want to sort another List ( 1 - continue or"//
               "0 - Exit) : ";
   cin>>option;

  }while(option!=0);

 cout<<"\n\t   END\n\n";
 return 0;
}
```

SAMPLE INPUT-OUTPUT



```
        MERGE SORT

Enter the number of terms in the List : 10
Enter Unsorted List elements to sort  : 23 45 67 89 12 11 6 -5 0 3

PASS : 1  [ 23 45 67 89 -5 6 11 12 0 3 ]

PASS : 2  [ -5 0 3 6 11 12 23 45 67 89 ]

        SORTED LIST

-5 0 3 6 11 12 23 45 67 89


Do you want to sort another List ( 1 - continue or 0 - Exit) : 1

Enter the number of terms in the List : 10
Enter Unsorted List elements to sort  : 23 98 88 12 70 22 0 52 16 20

PASS : 1  [ 12 23 88 98 0 22 52 70 16 20 ]

PASS : 2  [ 0 12 16 20 22 23 52 70 88 98 ]

        SORTED LIST

0 12 16 20 22 23 52 70 88 98


Do you want to sort another List ( 1 - continue or 0 - Exit) : 0

           END
```

SUBSTRING EXTRACTION

AIM

T understand the extraction string pattern from the given string, when the POS-position to start extraction and LEN - length from the substring to be extracted is given by the user.

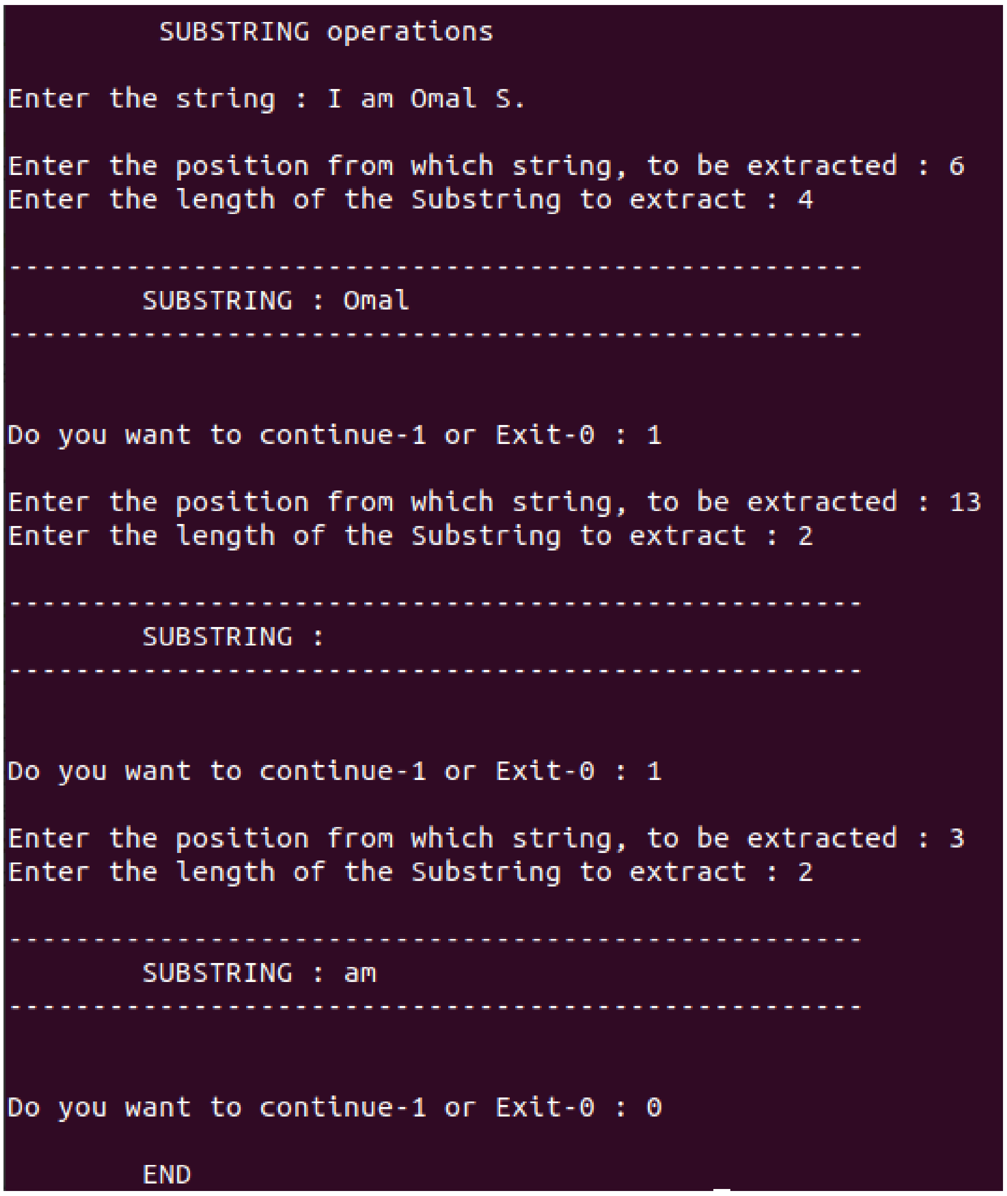
ALGORITHM

```
    SUBSTRING ( S , P , P_L)
        1.  Set N = S.length()
        2.  Create STR[N+1]
        3.  Create SUB[P_L]
        4.  Repeat the step for COUNT=0 to COUNT<P_L then
                (i) Set SUB[COUNT] = S[P_L + COUNT - 1]
        5.  End for loop
        6.  Set SUB[COUNT] = "\0"
        7.  print SUB
        8.  Exit
```

PROGRAM

```
#include <iostream>

#include<cstring>

using namespace std;

void SUBSTRING( string S , int E_P , int E_L)
{
 int n = S.length();

 char str[n+1];

 strcpy(str,S.c_str());

 char SUB[E_L];

 int count;

 for(count=0;count<E_L;count++)
 {
  SUB[count] = S[E_P+count-1];
 }

 SUB[count]= '\0';
 cout<<"\n\tSUBSTRING : "<<SUB;
 }

int main()
{
 int n,E_POS,E_LEN,choice=1,P,length, del_i,DS,REM,POS,ch;

 cout<<"\n\t SUBSTRING operations \n";

 string S,SubS , SAM;

 cout<<"\nEnter the string : ";

 getline(cin,S);

 do
 {
    cout<<"\n";
  cout<<"Enter the position from which string, to be extracted : ";
cin>>E_POS;

cout<<"Enter the length of the Substring to extract : ";
cin>>E_LEN;
cout<<"\n---------------------------------------------------";

SUBSTRING(S , E_POS ,E_LEN);

cout<<"\n---------------------------------------------------\n\n";
    cout<<"\nDo you want to continue-1 or Exit-0 : ";

    cin>>choice;

  }while(choice!=0);

 cout<<"\n\tEND\n";
 return 0;
}
```

SAMPLE INPUT-OUTPUT



```
         SUBSTRING operations

Enter the string : I am Omal S.

Enter the position from which string, to be extracted : 6
Enter the length of the Substring to extract : 4

---------------------------------------------------
        SUBSTRING : Omal
---------------------------------------------------


Do you want to continue-1 or Exit-0 : 1

Enter the position from which string, to be extracted : 13
Enter the length of the Substring to extract : 2

---------------------------------------------------
        SUBSTRING :
---------------------------------------------------


Do you want to continue-1 or Exit-0 : 1

Enter the position from which string, to be extracted : 3
Enter the length of the Substring to extract : 2

---------------------------------------------------
        SUBSTRING : am
---------------------------------------------------


Do you want to continue-1 or Exit-0 : 0

        END
```

PATTERN MATCHING

AIM

to understand the operation on string to find the match in the pattern input by the user with the given String.

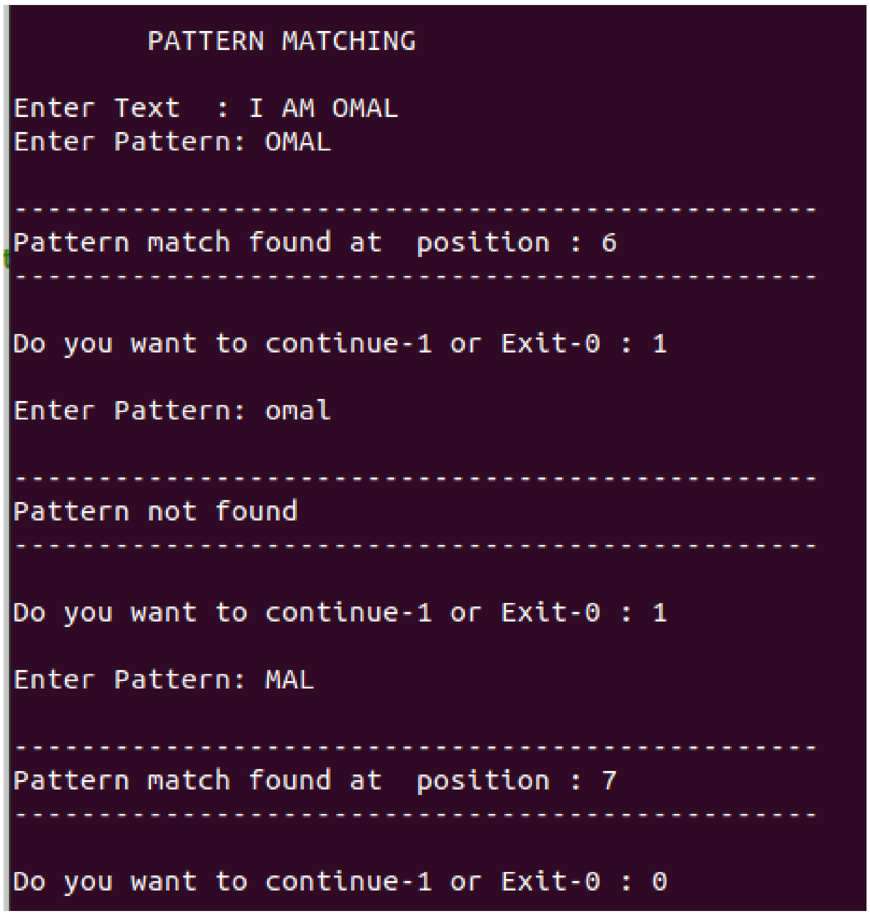
ALGORITHM

```
    PATTERN MATCHING (S , SUB, S_L , SUB_L)
        1.  Set MAXLEN = S_L-SUB_L +1
        2.  Set K = 0
        3.  Repeat the steps while K <= MAXLEN then
                (i)     Set C = 0
                (ii)    Repeat the steps for i = 0 to SUB_L then
                            (a) if SUB[i] != S[ K + i] then
                                *   i = SUB_L
                            (b) else
                                *   C=C+1
                            (c) End if condition
                (iii)   End for loop
                (iv)    if C == SUB_L then
                            (a) RETURN K
                (v)     End if condition
                (vi)    K = K+1
        4.  RETURN -1
        5.  Exit
```

PROGRAM

```
#include<iostream>
#include<cstring>
using namespace std;
int patternMatching(char T[], int tLength, char P[], int pLength)
{
    int k = 0;
    int maxLength = tLength - pLength + 1;
    while (k <= maxLength)
    {
        int c = 0;
        for (int i = 0; i < pLength; i++)
        {
            if (P[i] != T[k + i])
            {
                i = pLength;
            }
            else
            {
                c++;
            }
        }
        if (c == pLength)
        {
            return k;
        }
        k++;
    }
    return -1;
}
int main()
{

    string T1;
    char P[30];
    int tlen, plen, x;
    int option = 1;
    cout<<"\n\tPATTERN MATCHING\n";

    cout << "\nEnter Text  : ";
        getline(cin,T1);
        int n=T1.length();
        char T[n+1];
        strcpy(T , T1.c_str());



    do {

        cout << "Enter Pattern: ";
        cin >> P;
        tlen = strlen(T);
        plen = strlen(P);

        cout << "\n";
        x = patternMatching(T, tlen, P, plen) + 1;
        cout<<"------------------------------------------------\n";
        if (x == 0)
        {
            cout << "Pattern not found" << endl;
        }
        else
        {
            cout << "Pattern match found at  position : " << x << endl;
        }
        cout<<"------------------------------------------------\n";
        cout << "\n";
        cout << "Do you want to continue-1 or Exit-0 : ";
        cin >> option;
        cout << "\n";
    } while (option != 0);
    return 0;
}
```

SAMPLE INPUT-OUTPUT



```
        PATTERN MATCHING

Enter Text  : I AM OMAL
Enter Pattern: OMAL

------------------------------------------------
Pattern match found at  position : 6
------------------------------------------------

Do you want to continue-1 or Exit-0 : 1

Enter Pattern: omal

------------------------------------------------
Pattern not found
------------------------------------------------

Do you want to continue-1 or Exit-0 : 1

Enter Pattern: MAL

------------------------------------------------
Pattern match found at  position : 7
------------------------------------------------

Do you want to continue-1 or Exit-0 : 0
```

OPERATIONS ON STRING

AIM

To understand the operations on the STRING to be performed to REPLACEMENT, INSERT() and DELETES() sub-string in the given string

ALGORITHM

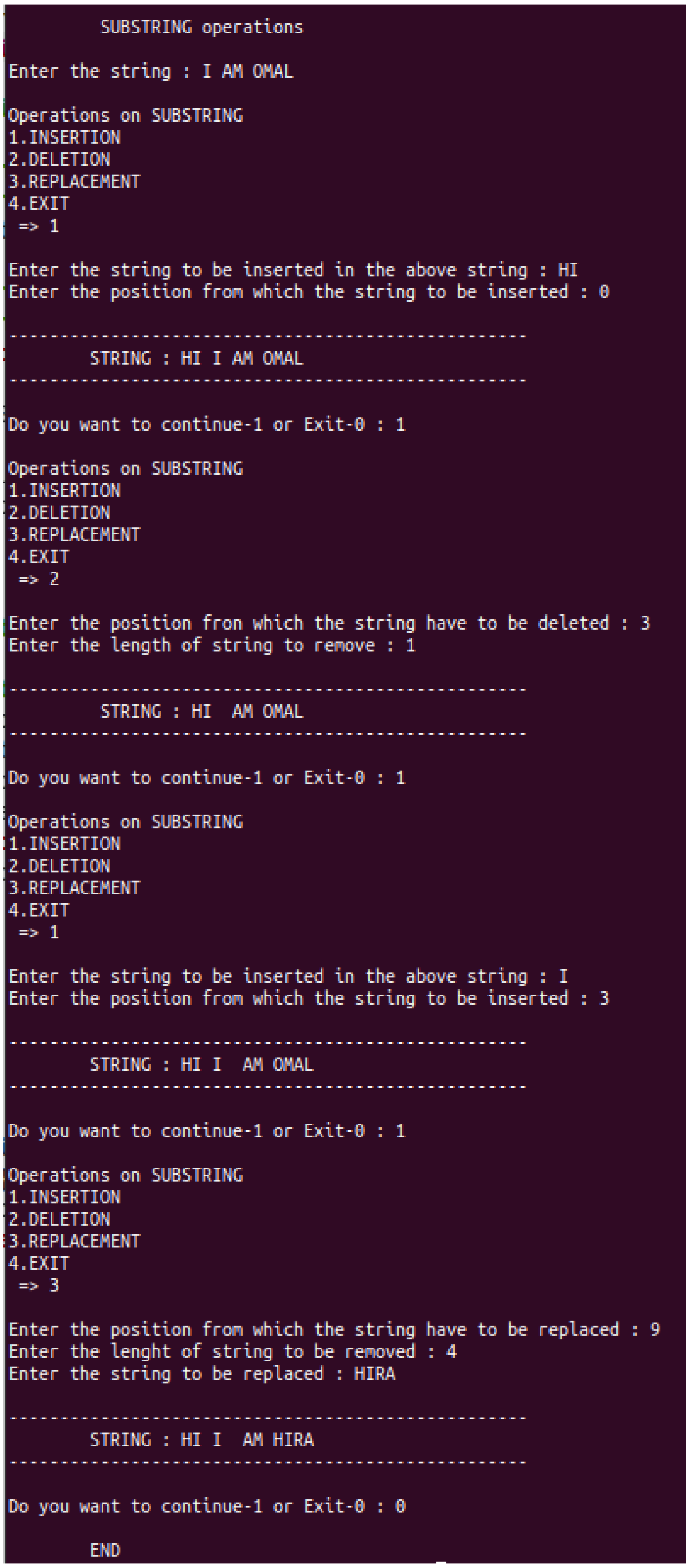
```
INSERT ( S , SUB , POS)
    1.  Set N = S.LENGTH()
    2.  Set NS = SUB.LENGTH()
    3.  Set TOTAL = N+NS+1
    4.  Create TEMP[TOTAL]
    5.  Repeat the step from i = POS to i< (NS+POS) then
        (i) Set STR[i] = SUB[i]
    6.  End for loop
    7.  Repeat the step from i = POS+NS to N+NS+1 then
        (i) Set STR[i] = TEMP[j]
    8.  End for loop
    9.  Exit

DELETE( S,POS , LEN )
    1.  Set NM = POS+LEN
    2.  Set N = S.LENGTH()
    3.  Create STR[N+1]
    4.  Set STR = S
    5.  Create TEMP[N+1]
    6.  Repeat the step for i=0 to POS then
        (i)     Set STR[i] = TEMP
        (ii)    Set NM = NM+1
    7.  End for loop
    8.  RETURN STR
    9.  Exit
```

PROGRAM

```
#include <iostream>
#include<cstring>
using namespace std;
```

```
string insert(string S, int E_p, string SUB)
 {
  int n=S.length();
  SUB =SUB+" ";
  int ns = SUB.length();
  int total = n+ns+1;
  char str[total];
  strcpy(str,S.c_str());
  char Sub[ns+1];
  strcpy(Sub , SUB.c_str());

  string tep=S;

  char temp[total];
  strcpy(temp , tep.c_str());

  for(int i=E_p,j=0;i<ns+E_p;i++,j++)
  {
   str[i] = Sub[j];
  }

  for(int i=E_p+ns,j=E_p;i<n+ns+1;i++,j++)
  {
   str[i] = temp[j];
  }
  cout<<"STRING : "<<str<<"\n";
  return(string(str));
 }

 string deleteS(string S, int pos , int len )
 {
  int nm = pos+len;
  int n = S.length();
  char str[n+1];
  strcpy(str,S.c_str());

   string tep=S;
   char temp[n+1];
  strcpy(temp , tep.c_str());
  for(int i=0,j=pos;j<n+1;i++)
  {
   str[j]=temp[nm];
   nm =nm+1;
   j=j+1;
  }
  return (string(str));
 }

int main()
{
int n,E_POS,E_LEN,choice=1,P,length, del_i,DS,REM,POS,ch;
 cout<<"\n\t SUBSTRING operations \n";
 string S,SubS , SAM;
 cout<<"\nEnter the string : ";
 getline(cin,S);
 do
 {
 cout<<"\nOperations on SUBSTRING\n1.INSERTION\n2.DELETION\n3.REPLACEMENT"//
                   "\n4.EXIT\n => ";
 cin>>ch;
 switch(ch)
  {
     case 1:
     {
     cout<<"\nEnter the string to be inserted in the above string : ";
    cin>>SubS;
    cout<<"Enter the position from which the string to be inserted : ";
    cin>>P;
   cout<<"\n---------------------------------------------------\n\t";

    S = insert(S , P,SubS);

    cout<<"---------------------------------------------------\n";
     break;
    }
    case 2:
     {
     cout<<"\nEnter the position fron which the string have to be deleted : ";
    cin>>del_i;
    cout<<"Enter the length of string to remove : ";
    cin>>DS;
    S = deleteS(S , del_i , DS);

    cout<<"\n---------------------------------------------------";
    cout<<"\n\t STRING : "<<S;
    cout<<"\n---------------------------------------------------\n";
    break;
    }
    case 3:
    {

     cout<<"\nEnter the position from which the string have to be replaced : ";
    cin>>POS;
    cout<<"Enter the lenght of string to be removed : ";
    cin>>REM;
    cout<<"Enter the string to be replaced : ";
    cin>>SAM;
    cout<<"\n---------------------------------------------------\n\t";

    S = deleteS(S , POS , REM);
    S = insert(S , POS ,SAM);

    cout<<"---------------------------------------------------\n";
    break;
    }
    case 4:
    {
     choice = 0;
     break;
    }
    default:cout<<"\nWrong choice\n";
  }
 if(choice!=0)
 {
    cout<<"\nDo you want to continue-1 or Exit-0 : ";
    cin>>choice;
 }

}while(choice!=0);
cout<<"\n\tEND\n";
return 0;
}
```

SAMPLE INPUT-OUTPUT



```
        SUBSTRING operations

Enter the string : I AM OMAL

Operations on SUBSTRING
1.INSERTION
2.DELETION
3.REPLACEMENT
4.EXIT
 => 1

Enter the string to be inserted in the above string : HI
Enter the position from which the string to be inserted : 0

---------------------------------------------------
       STRING : HI I AM OMAL
---------------------------------------------------

Do you want to continue-1 or Exit-0 : 1

Operations on SUBSTRING
1.INSERTION
2.DELETION
3.REPLACEMENT
4.EXIT
 => 2

Enter the position fron which the string have to be deleted : 3
Enter the length of string to remove : 1

---------------------------------------------------
        STRING : HI  AM OMAL
---------------------------------------------------

Do you want to continue-1 or Exit-0 : 1

Operations on SUBSTRING
1.INSERTION
2.DELETION
3.REPLACEMENT
4.EXIT
 => 1

Enter the string to be inserted in the above string : I
Enter the position from which the string to be inserted : 3

---------------------------------------------------
       STRING : HI I  AM OMAL
---------------------------------------------------

Do you want to continue-1 or Exit-0 : 1

Operations on SUBSTRING
1.INSERTION
2.DELETION
3.REPLACEMENT
4.EXIT
 => 3

Enter the position from which the string have to be replaced : 9
Enter the lenght of string to be removed : 4
Enter the string to be replaced : HIRA

---------------------------------------------------
       STRING : HI I  AM HIRA
---------------------------------------------------

Do you want to continue-1 or Exit-0 : 0

       END
```

SPARSE MATRIX TRANSPOSE AND ADDITION

AIM

To understand the addition and transpose of the sparse matrix using sparse table .

ALGORITHM

```
 ADD ( MATRIX1 , MATRIX2)
    1.  if MATRIX1.ROW1 != MATRIX2.ROW2 then
            (i)     print "CANNOT BE ADDED";
    2.  else
            (i)     Set APOS = 0
            (ii)    Set BPOS = 0
            (iii)    Create NEW Node
            (iv)     Repeat the step while
                        (APOS < MATRIX1.LEN && BPOS < MATRIX2.LEN)

                    (a) if MATRIX1.DATA[APOS][0] > MATRIX2.DATA[BPOS][0]
                    || MATRIX1.DATA[APOS][0] == MATRIX2.DATA[BPOS][0]
                    && MATRIX1.DATA[APOS][1] > MATRIX2.DATA[BPOS][1]))

                        *   RESULT.INSERT(MATRIX2.DATA[BPOS][0],
                        MATRIX2.DATA[BPOS][1],
                        MATRIX.DATA[BPOS][2]);
                        *   BPOS++;

                    (b) else if (MATRIX1.DATA[APOS][0] < MATRIX2.DATA[BPOS][0] ||
                    (MATRIX.DATA[APOS][0] == MATIRX2.DATA[BPOS][0]
                    && MATRIX1.DATA[APOS][1] < MATRIX2.DATA[BPOS][1]))
                        *   RESULT.INSERT(MATRIX1.DATA[APOS][0],
                        MATRIX1.DATA[APOS][1],
                        MATRIX1.DATA[APOS][2]);
                        *   APOS = APOS+1;

                    (c) else
                        * Set ADDEDVAL = MATRIX.DATA[APOS][2] +
                        MATRIX2.DATA[BPOS][2]
                        *   if (ADDEDVAL != 0)
                                # RESULT.INSERT(MATRIX.DATA[APOS][0],
                                MATRIX1.DATA[APOS][1],ADDEDVAL)
                                #   APOS = APOS+1
```

```
                            #   BPOS = BPOS+1
                        * End if condition
                    (d) End if condition
            (v)  End while loop
            (vi) Repeat the steps while APOS < LEN then
                    (a)  RESULT.INSERT(MATRIX1.DATA[APOS][0],
                        MATRIX1.DATA[APOS][1],
                        MATRIX1.DATA[APOS++][2]
            (vii) End while loop
            (viii) Repeat the steps while BPOS < LEN then
                    (a)   RESULT.INSERT(MATRIX2.DATA[BPOS][0],
                    MATRIX2.DATA[BPOS][1],
                    MATRIX2.DATA[BPOS++][2]
            (ix) End while loop
            (x) RESULT.PRINT()
    3.  End if condition
    4.  Exit

    TRANSPOSE ( MATRIX)
        1.  Set RESULT(MATRIX.COL, MATRIX.ROW)
        2.  Set RESULT.LEN = LEN
        3.  Create COUNT matrix of size MATRIX.COL+1
        4.  Repeat the step i=1 to MATRIX.COL then
                (i) Set COUNT[i] = 0
        5.  Repeat the step i=0 to LEN then
                (i) Set COUNT[MATRIX.DATA[i][1]]++
        6.  End for loop
        7.  Create INDEX matrix of size col+1
        8.  Set INDEX[0] =0
        9.  Repeat the steps i=1 to col then
                (i) Set INDEX[i] = INDEX[i-1] + COUNT[i-1]
        10. End for loop
        11. Repeat the step for i=0 to i<LEN then
                (i)     Set RPOS = INDEX[DATA[i][1]]++
                (ii)    Set RESULT.DATA[RPOS][0] = MATRIX.DATA[i][1]
                (iii)   Set RESULT.DATA[RPOS][1] = MATRIX.DATA[i][0]
                (iv)    Set RESULT.DATA[RPOS][2] = MATRIX.DATA[i][2]
        12. End for loop
        13. RETURN
        14. Exit
```

PROGRAM

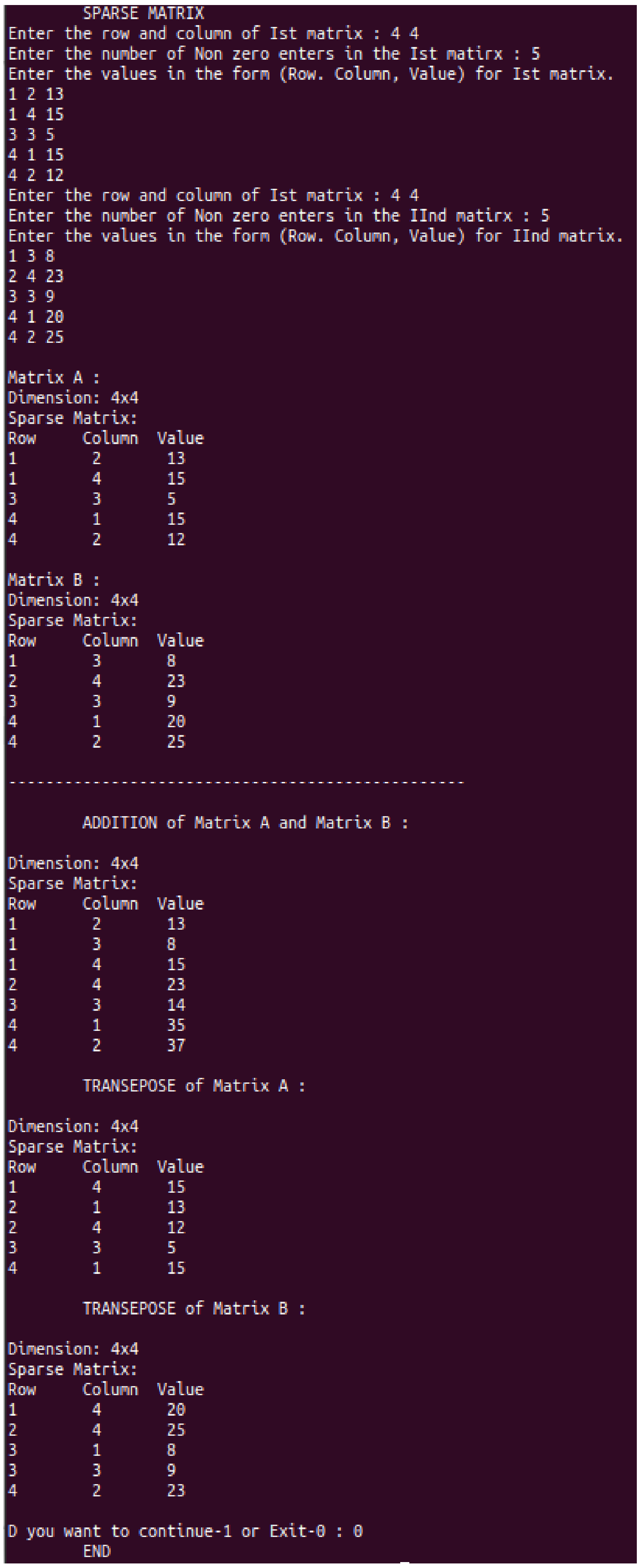
```
#include <iostream>
using namespace std;

class sparse_matrix
{
const static int MAX = 100;
int **data;
int row, col;
int len;
public:
sparse_matrix(int r, int c)
{
row = r;
col = c;
len = 0;
data = new int *[MAX];
for (int i = 0; i < MAX; i++)
{
data[i] = new int[3];
}
}
void insert(int r, int c, int val)
{
if (r > row || c > col)
{
cout << "Wrong entry";
}
else
{
data[len][0] = r;
data[len][1] = c;
data[len][2] = val;
len = len+1;
}
}
void add(sparse_matrix b)
```

```
{
if (row != b.row || col != b.col)
{
cout << "Matrices can't be added";
}

else
{
int apos = 0, bpos = 0;
sparse_matrix result(row, col);

while (apos < len && bpos < b.len)
{

if (data[apos][0] > b.data[bpos][0] ||
(data[apos][0] == b.data[bpos][0] &&
data[apos][1] > b.data[bpos][1]))

{
result.insert(b.data[bpos][0],
b.data[bpos][1],
b.data[bpos][2]);

bpos = bpos+1;
}

else if (data[apos][0] < b.data[bpos][0] ||
(data[apos][0] == b.data[bpos][0] &&
data[apos][1] < b.data[bpos][1]))

{
result.insert(data[apos][0],
data[apos][1],
data[apos][2]);

apos = apos+1;
}

else
{
int addedval = data[apos][2] +
b.data[bpos][2];

if (addedval != 0)
{
result.insert(data[apos][0],
data[apos][1],
addedval);
}
apos = apos+1;
bpos = bpos+1;
}
}
while (apos < len)
{ result.insert(data[apos][0],
data[apos][1],
data[apos++][2]);
}

while (bpos < b.len)
{
result.insert(b.data[bpos][0],
b.data[bpos][1],
b.data[bpos++][2]);
            }
result.print();
}
}

sparse_matrix transpose()
{
sparse_matrix result(col, row);
result.len = len;
int *count = new int[col + 1];
for (int i = 1; i <= col; i++)
{
    count[i] = 0;
}

for (int i = 0; i < len; i++)
{
    count[data[i][1]]++;
```

```
}

int *index = new int[col + 1];
index[0] = 0;
for (int i = 1; i <= col; i++)
        {
index[i] = index[i - 1] + count[i - 1];
        }
for (int i = 0; i < len; i++)
{
int rpos = index[data[i][1]]++;
result.data[rpos][0] = data[i][1];
result.data[rpos][1] = data[i][0];
result.data[rpos][2] = data[i][2];
}

return result;
}

void print()
{
cout << "\nDimension: " << row << "x" << col;
cout << "\nSparse Matrix: \nRow\tColumn\tValue\n";

for (int i = 0; i < len; i++)
{
cout << data[i][0] << "\t " << data[i][1]
<< "\t " << data[i][2] << endl;
}
}
};
int main()
{
    cout<<"\n\tSPARSE MATRIX\n";
int A_m[2], B_m[2], option;
int A[3],B[3];
int count;
do
{
cout<<"Enter the row and column of Ist matrix : ";
```

```
for(int i=0;i<2;i++)
{
 cin>>A_m[i];
}
sparse_matrix a(A_m[0] , A_m[1]);

cout<<"Enter the number of Non zero enters in the Ist matirx : ";

cin>>count;

cout<<"Enter the values in the form (Row. Column, Value) for Ist matrix.\n";

for(int i = 0;i<count;i++)
{
  for(int j=0;j<3;j++)
  {
    cin>>A[j];
  }
  a.insert(A[0], A[1], A[2]);
}

cout<<"Enter the row and column of Ist matrix : ";

for(int i=0;i<2;i++)
{
 cin>>B_m[i];
}

sparse_matrix b(B_m[0] , B_m[1]);

cout<<"Enter the number of Non zero enters in the IInd matirx : ";

cin>>count;

cout<<"Enter the values in the form (Row. Column, Value) for IInd matrix.\n";

for(int i = 0;i<count;i++)
{
  for(int j=0;j<3;j++)
  {
    cin>>B[j];
  }
  b.insert(B[0], B[1], B[2]);
}

cout<<"\nMatrix A : ";

a.print();

cout<<"\nMatrix B : ";

b.print();

cout<<"\n-------------------------------------------------\n";

cout << "\n\tADDITION of Matrix A and Matrix B : \n";

a.add(b);

cout<<"\n\tTRANSEPOSE of Matrix A : \n";

sparse_matrix atranspose = a.transpose();

atranspose.print();

cout<<"\n\tTRANSEPOSE of Matrix B : \n";

sparse_matrix btranspose = b.transpose();

btranspose.print();

cout<<"\nDo you want to continue-1 or Exit-0 : ";

cin>>option;

}while(option != 0);

cout<<"\tEND\n";

return 0;
}
```

SAMPLE INPUT-OUTPUT



```
        SPARSE MATRIX
Enter the row and column of Ist matrix : 4 4
Enter the number of Non zero enters in the Ist matirx : 5
Enter the values in the form (Row. Column, Value) for Ist matrix.
1 2 13
1 4 15
3 3 5
4 1 15
4 2 12
Enter the row and column of Ist matrix : 4 4
Enter the number of Non zero enters in the IInd matirx : 5
Enter the values in the form (Row. Column, Value) for IInd matrix.
1 3 8
2 4 23
3 3 9
4 1 20
4 2 25

Matrix A :
Dimension: 4x4
Sparse Matrix:
Row     Column  Value
1        2       13
1        4       15
3        3       5
4        1       15
4        2       12

Matrix B :
Dimension: 4x4
Sparse Matrix:
Row     Column  Value
1        3       8
2        4       23
3        3       9
4        1       20
4        2       25

-----------------------------------------------

        ADDITION of Matrix A and Matrix B :

Dimension: 4x4
Sparse Matrix:
Row     Column  Value
1        2       13
1        3       8
1        4       15
2        4       23
3        3       14
4        1       35
4        2       37

        TRANSEPOSE of Matrix A :

Dimension: 4x4
Sparse Matrix:
Row     Column  Value
1        4       15
2        1       13
2        4       12
3        3       5
4        1       15

        TRANSEPOSE of Matrix B :

Dimension: 4x4
Sparse Matrix:
Row     Column  Value
1        4       20
2        4       25
3        1       8
3        3       9
4        2       23

D you want to continue-1 or Exit-0 : 0
        END
```

SADDLE POINT

AIM

To understand the algorithm to find the saddle point in a given matrix.

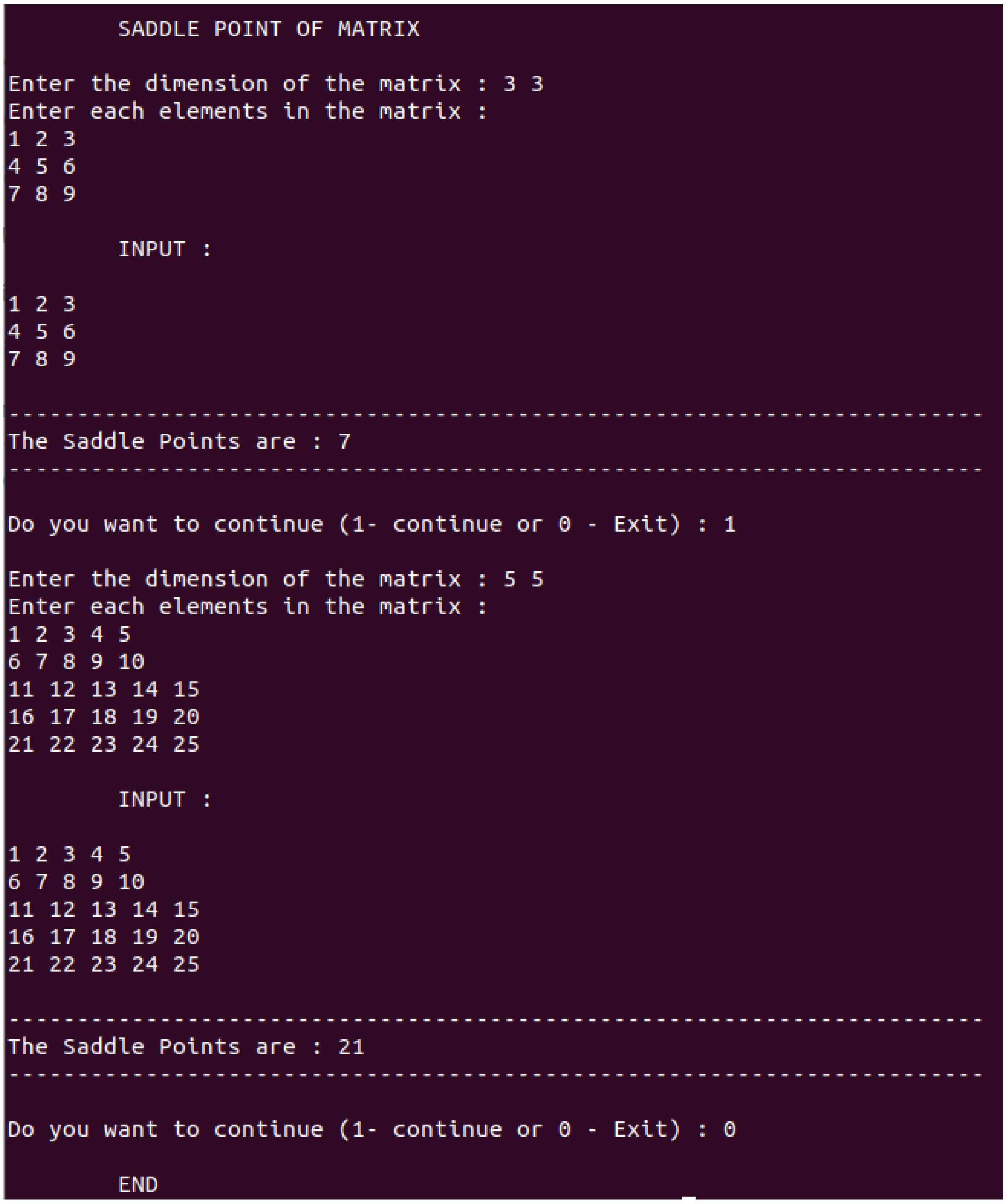
ALGORITHM

```
SADDLEPOINT( MATRIX )
    1.  Create the M1[DIM1] AND M2[DIM2]
    2.  Repeat the the steps for i=1 to i < DIM1
            (i)   M1[i] = MATRIX[i][0]
            (ii)  M2[i] = MATRIX[0][i]
            (iii) Repeat the steps for j=0 to DIM2
                    (a) if M1[i] > MATRIX[i][j] then
                            *  Set M1[i] = MATRIX[i][j]
                    (b) End if condition
                    (c) if M2[i] = MATRIX[j][i]
                            *   M2[i] = MATRIX[j][i]
                    (d) End if condition
            (iv)  End four loop
    3.  End for loop
    4.  print the saddle points stored in M1[DIM]
    5.  RETURN 0
    6.  Exit
```

PROGRAM

```
#include<iostream>

using namespace std;

class matrix
{
int **Matrix;
int dimX;
int dimY;
public:
 matrix(){};
 matrix(int x,int y)
 {
  dimX=x;
  dimY=y;
  Matrix=new int *[dimX];
  for(int i=0;i<dimX;i++)
    {
     Matrix[i]=new int [dimY];
    }
 }

 void input(int &i,int &j,int value=0)
  {
  Matrix[i][j]=value;
  }

int SADDLE_POINT()
{
 int m[dimX];
 int n[dimY];
 int i,j;
 int c = 1;
 for( i=0;i<dimX;i++)
 {
  m[i] = Matrix[i][0];
  n[i] = Matrix[0][i];

  for( j=0;j<dimY;j++)
  {
   if (m[i] > Matrix[i][j] )
   {
    m[i] = Matrix[i][j];
   }
   if(n[i] < Matrix[j][i])
   {
    n[i] = Matrix[j][i];
   }
  }
 }

 cout<<"\nThe Saddle Points are : ";

  for(i =0;i<dimX;i++)
  {
   for(j=0;j<dimY;j++)
   {
    if(m[i] == n[j])
    {
     cout<<m[i]<<"  ";
     c = 0;
    }
   }
  }
  cout<<endl;
  return c;
}

void Display()
{
 for(int i=0;i<dimX;i++)
 {
 for(int j=0;j<dimY;j++)
 {
  cout<<Matrix[i][j]<<" ";
 }
 cout<<"\n";
 }
}

~matrix()
  {
    for(int i=0;i<dimX;i++)
    {delete Matrix[i];}
    delete Matrix;
  }
};

int main()
{
 cout<<"\n\tSADDLE POINT OF MATRIX\n";
 int dimX , dimY, element,choice;

 do
 {
 cout<<"\nEnter the dimension of the matrix : ";
 cin>>dimX>>dimY;
 matrix M(dimX,dimY);
 cout<<"Enter each elements in the matrix : \n";
 for(int i=0;i<dimX;i++)
 {
  for(int j = 0; j<dimY;j++)
  {
   cin>>element;
   M.input(i,j,element);
  }
 }
 cout<<"\n\tINPUT : \n\n";
 M.Display();
 cout<<"\n-----------------------------------------------------------------------";

 if ( M.SADDLE_POINT() == 1)
 {
  cout<<"\nThe matrix have no saddle point. \n";
 }
 cout<<"-----------------------------------------------------------------------\n";
 cout<<"\nDo you want to continue (1- continue or 0 - Exit) : ";
 cin>>choice;

 }while(choice != 0);
 cout<<"\n\tEND\n";
 return 0;
}
```

SAMPLE INPUT-OUTPUT



```
        SADDLE POINT OF MATRIX

Enter the dimension of the matrix : 3 3
Enter each elements in the matrix :
1 2 3
4 5 6
7 8 9

        INPUT :

1 2 3
4 5 6
7 8 9

-----------------------------------------------------------------------
The Saddle Points are : 7
-----------------------------------------------------------------------

Do you want to continue (1- continue or 0 - Exit) : 1

Enter the dimension of the matrix : 5 5
Enter each elements in the matrix :
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25

        INPUT :

1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25

-----------------------------------------------------------------------
The Saddle Points are : 21
-----------------------------------------------------------------------

Do you want to continue (1- continue or 0 - Exit) : 0

        END
```

POLYNOMIAL ADDITION USING ARRAY

AIM

To understand the addition of polynomial numbers using the ADD() function in with the Array data structure is used.

ALGORITHM

```
ADD( POLY1 , POLY2)
    1.  Set S_A = 0
    2.  Set S_B = 0
    3.  Set S_C = 0
    4.  if (size != 0)
            (i)  Repeat the steps while S_A < A.SIZE or S_B < B.SIZE theN
                  (a) if A.POWER[S_A] < B.POWER[S_B]  OR A.POWER[S_A] > B.POWER[S_B]
                            * if A.POWER[S_A] < B.POWER[S_B]
                                   #   Set POWER[S_C] = A.POWER[S_A]
                                   #   Set CONSTANT[S_C] = A.CONSTANT[S_A]
                                   #   Set S_A = S_A+1
                                   #   Set S_C = S_C+1
                            *   else
                                #   Set POWER[S_C] = A.POWER[S_B]
                                #   Set CONSTANT[S_C] = A.CONSTANT[S_B]
                                #   Set S_B = S_B+1
                                #   Set S_C = S_C+1
                            *   End if condition
                   (b)else
                            *   Set POWER[S_C] = A.POWER[S_A]
                            *   Set CONSTANT[S_C] = A.CONSTANT[S_A] + B.CONSTANT[S_B]
                            *   S_A = S_A+1
                            *   S_B = S_B+1
                            *   S_C = S_C+1
                    (c) End if condition
            (ii)  End while loop
            (iii) if S_A < A.SIZE Then
                    (a) Repeat the steps while S_B < B_SIZE then
                            *   Set POWER[S_C] = B.POWER[S_B]
                            *   Set CONSTANT[S_C] = B.CONSTANT[S_B]
                            *   Set S_C = S_C + 1
                            *   Set S_B = S_B + 1
                    (b) End for loop
            (iv)  else
                    (a) Repeat the steps for S_A < A.SIZE then
                            *   Set POWER[S_C] = A. POWER[S_A]
                            *   Set CONSTANT[S_C] = A.CONSTANT[S_A]
                            *   Set S_C = S_C+1
                            *   Set S_A = S_A+1
            (v)     End if condition
    5. else
            (i) print "SIZE OF POLYNOMIAL IS ZERO";
    6. End if condition
    7. Exit
```

PROGRAM

```
#include <iostream>
using namespace std;

class POLY
{
 public:
 int *power;
 int *constant;
 int size;

   POLY(int S)
   {
     power = new int[S];
     constant= new int [S];
     size = S;
   }
   void print(int S)
   {
    cout<<constant[0]<<"x^"<<power[0]<<" ";
    for(int i=1;i<S;i++)
    {
     if(constant[i]>=0)
     {  cout<<"+";  }
     cout<<" "<<constant[i]<<"x^"<<i<<" ";
    }
    cout<<"\n";
   }
```

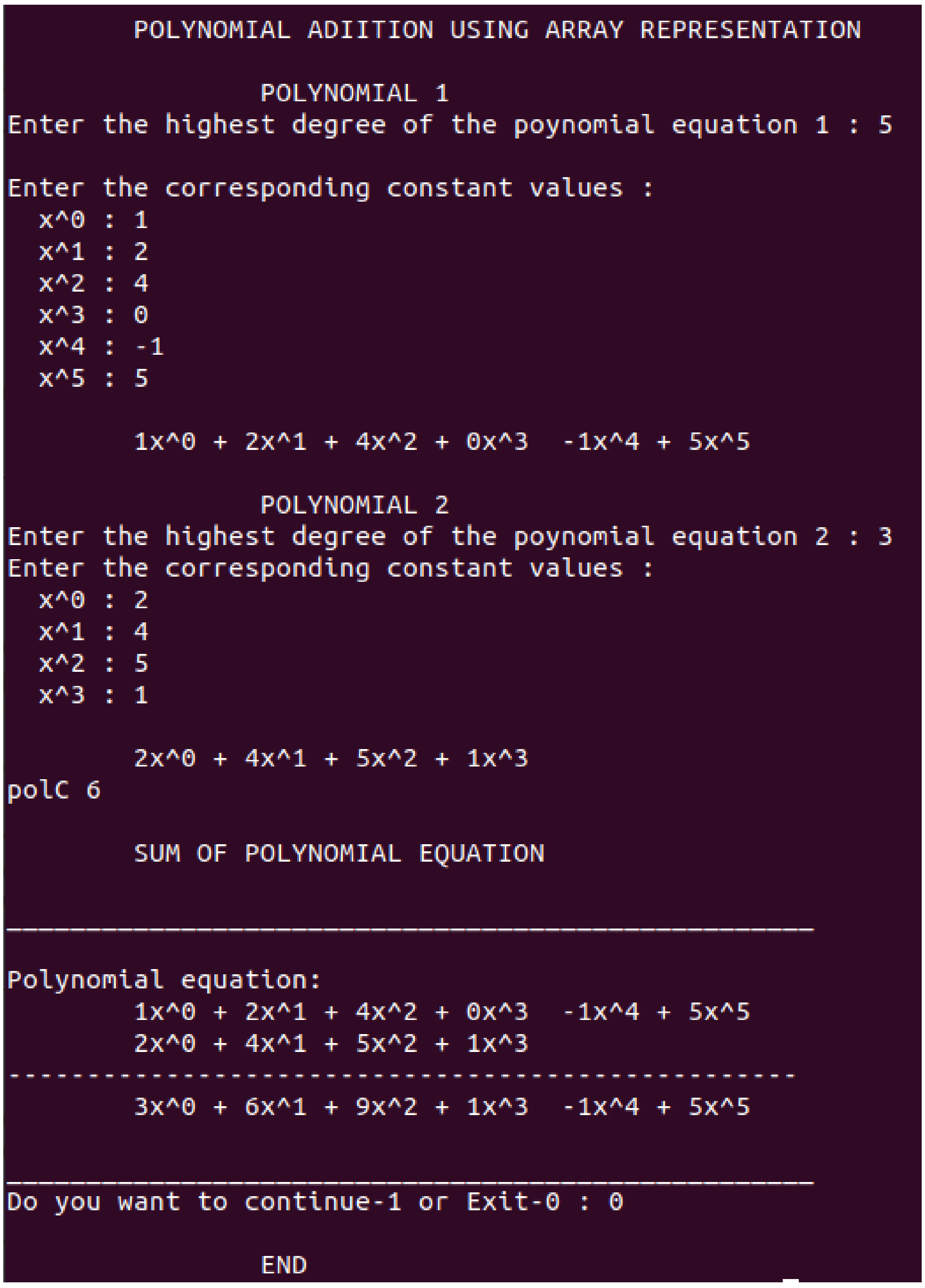
```
void add(POLY A, POLY B)
{
  cout<<"\n\tSUM OF POLYNOMIAL EQUATION \n";
  int s_A = 0;
  int s_B = 0;
  int s_C = 0;
  if (size!=0)
  {
    while((s_A<A.size) || (s_B<B.size))
    {
     if((A.power[s_A] < B.power[s_B]) || (A.power[s_A] > B.power[s_B]))
     {
      if (A.power[s_A] < B.power[s_B])
      {
       power[s_C] = A.power[s_A];
       constant[s_C] = A.constant[s_A];
       s_A = s_A+1;
       s_C = s_C+1;
      }
      else
      {
       power[s_C] = B.power[s_B];
       constant[s_C] = A.constant[s_B];
       s_B = s_B+1;
       s_C = s_C+1;
      }
     }
     else
     {
      power[s_C] = A.power[s_A];
      constant[s_C] = A.constant[s_A] + B.constant[s_B];
      s_A = s_A+1;
      s_B = s_B+1;
       s_C = s_C+1;
     }
    }
    if(s_A<A.size)
    {
     for(s_B;s_B<B.size;)
     {
       power[s_C] = B.power[s_B];
```

```
          constant[s_C] = B.constant[s_B];
          s_C = s_C+1;
          s_B = s_B+1;
        }
       }
       else
       {
        for(s_A;s_A<A.size;)
        {
          power[s_C] = A.power[s_A];
          constant[s_C] = A.constant[s_A];
          s_C = s_C+1;
          s_A = s_A+1;
        }
       }
      }
      else
      {
        cout<<"\nThe size of resultant polynomial is calculated to zero.\n ";
      }
   }

};

int main()
{
 cout<<"\n\tPOLYNOMIAL ADIITION USING ARRAY REPRESENTATION\n";
 int polA, polB, polC,option;
 do
 {
 cout<<"\n\t\tPOLYNOMIAL 1 \n";
 cout<<"Enter the highest degree of the poynomial equation 1 : ";
 cin>>polA;
 polA =polA+1;
 POLY A(polA);
 cout<<"\nEnter the corresponding constant values :\n";
 for(int i=0;i<polA;i++)
 {
  cout<<"  x^"<<i<<" : ";
  cin>>A.constant[i];
  A.power[i] = i;
```

```
}
cout<<"\n\t";
A.print(polA);

cout<<"\n\t\tPOLYNOMIAL 2 \n";
cout<<"Enter the highest degree of the poynomial equation 2 : ";
cin>>polB;
polB=polB+1;
POLY B(polB);
cout<<"Enter the corresponding constant values :\n";
for(int i=0;i<polB;i++)
{
 cout<<"  x^"<<i<<" : ";
 cin>>B.constant[i];
 B.power[i] = i;
}
cout<<"\n\t";
B.print(polB);

if(polA>polB)
{    polC = polA;    }
else
{    polC = polB;    }
cout<<"polC "<<polC<<endl;
POLY C(polC);

C.add( A, B);

cout<<"\n___________________________________________________\n";
cout<<"\nPolynomial equation: \n\t";

A.print(polA);

cout<<"\t";

B.print(polB);

cout<<"--------------------------------------------------\n\t";
C.print(polC);
cout<<"\n___________________________________________________";
```

```
 cout<<"\nDo you want to continue-1 or Exit-0 : ";
 cin>>option;
 }while(option != 0);
 cout<<"\n\t\tEND\n";
 return 0;
}
```

SAMPLE INPUT-OUTPUT



```
        POLYNOMIAL ADIITION USING ARRAY REPRESENTATION

                POLYNOMIAL 1
Enter the highest degree of the poynomial equation 1 : 5

Enter the corresponding constant values :
  x^0 : 1
  x^1 : 2
  x^2 : 4
  x^3 : 0
  x^4 : -1
  x^5 : 5

        1x^0 + 2x^1 + 4x^2 + 0x^3  -1x^4 + 5x^5

                POLYNOMIAL 2
Enter the highest degree of the poynomial equation 2 : 3
Enter the corresponding constant values :
  x^0 : 2
  x^1 : 4
  x^2 : 5
  x^3 : 1

        2x^0 + 4x^1 + 5x^2 + 1x^3
polC 6

        SUM OF POLYNOMIAL EQUATION

___________________________________________________

Polynomial equation:
        1x^0 + 2x^1 + 4x^2 + 0x^3  -1x^4 + 5x^5
        2x^0 + 4x^1 + 5x^2 + 1x^3
--------------------------------------------------
        3x^0 + 6x^1 + 9x^2 + 1x^3  -1x^4 + 5x^5


___________________________________________________
Do you want to continue-1 or Exit-0 : 0

                END
```

POLYNOMIAL ADDITION USING LINKED LIST

AIM

To understand the addition of polynomial using the values stored as each node of linked list.

ALGORITHM

```
ADDITION( POLY1 , POLY2)
    1.  Create TEMP1
    2.  Set TEMP1 = POLY1.HEAD
    3.  Create TEMP2
    4.  Set TEMP2 = POLY2.HEAD
    5.  Repeat the steps TEMP1 != NULL and TEMP2 != NULL then
            (i)     if TEMP1->POWER = TEMP2->POWER then
                        (a) Set SUM = TEMP->INFO + TEMP2->INFO
                        (b) Create NEW Node with SUM and TEMP1->POWER
                        (c) Set TEMP1 = TEMP1->LINK
                        (d) Set TEMP2 = TEMP2->LINK
            (ii)    else if TEMP1->POWER > TEMP2->POWER then
                        (b) Create NEW Node with TEMP1->INFO and TEMP1->POWER
                        (c) Set TEMP1 = TEMP1->LINK
            (iii)   else
                        (a) Create NEW Node with TEMP2->INFO and TEMP2->POWER
                        (b) Set TEMP2 = TEMP2->LINK
            (iv)    End if condition
    6.  End while loop
    7.  Repeat the steps while TEMP1 != NULL then
            (i)     Create NEW NODE with TEMP1->INFO and TEMP1->POWER
            (ii)    Set TEMP1 = TEMP1->LINK
    8.  End while loop
    9.  Repeat the steps while TEMP2 != NULL then
            (i)     Create NEW NODE with TEMP2->INFO and TEMP2->POWER
            (ii)    Set TEMP2 = TEMP2->LINK
    10. End while loop
    11. Exit
```

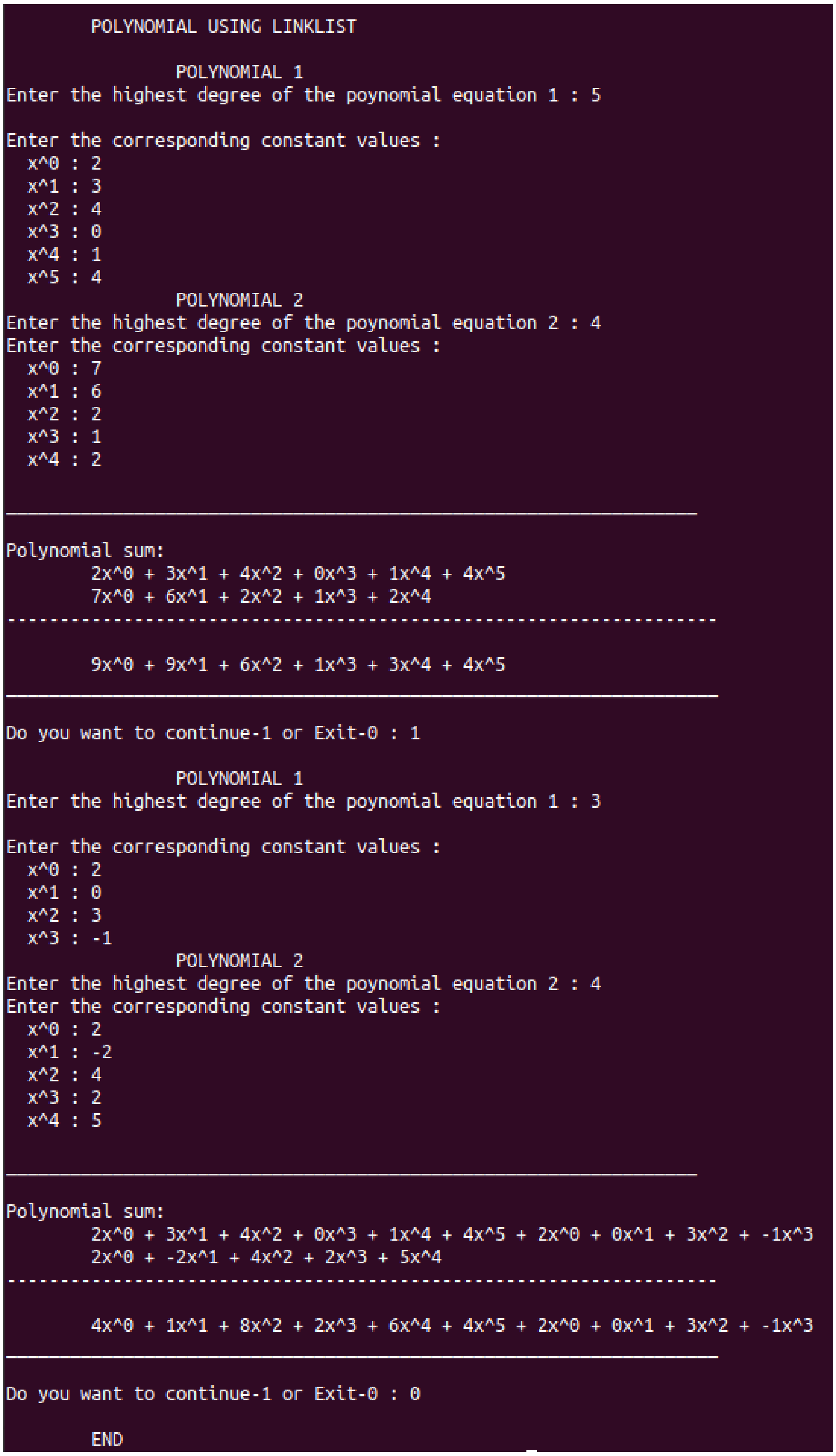
PROGRAM

```
#include <iostream>
using namespace std;
class Node
{
```

```
int coff,power;
Node* link;
public:
Node()
{
coff = 0;
power = 0;
link = NULL;
}
Node(int coff)
{
this->coff = coff;
power = 0;
link = NULL;
}
Node(int coff,int power)
{
this->coff = coff;
this->power = power;
link = NULL;
}
int getPower()
{
return this->power;
}
int getCoefficient()
{
return this->coff;
}
friend class LinkedList;
};
class LinkedList
{
Node* head;
public:
LinkedList()
{
head = NULL;
}
void newNode(int val,int pow);
void printPolynomial();
void addPolynomial(LinkedList poly1,LinkedList poly2);
};
void LinkedList::newNode(int val,int pow)
{
Node* newNode = new Node(val,pow);
if(head==NULL)
{
head = newNode;
return;
}
Node* temp = head;
while(temp->link!=NULL)
{
temp = temp->link;
}
temp->link = newNode;
}
void LinkedList::printPolynomial()
{
Node* temp = head;
while(temp!=NULL)
{
cout<<temp->coff<<"x^"<<temp->power;
if(temp->link!=NULL)
{
cout<<" + ";
}
temp = temp->link;
}
}
void LinkedList::addPolynomial(LinkedList p1,LinkedList p2)
{
Node* temp1 = p1.head;
Node* temp2 = p2.head;
while(temp1!=NULL and temp2!=NULL)
{
if(temp1->getPower()==temp2->getPower())
{
int sum = temp1->getCoefficient() + temp2->getCoefficient();
newNode(sum, temp1->getPower());
temp1 = temp1->link;
```

```
temp2 = temp2->link;
}
else if (temp1->getPower()>temp2->getPower())
{
newNode(temp1->getCoefficient(), temp1->getPower());
temp1 = temp1->link;
}
else
{
newNode(temp2->getCoefficient(), temp2->getPower());
temp2 = temp2->link;
}
}
while(temp1!=NULL)
{
newNode(temp1->getCoefficient(), temp1->getPower());
temp1 = temp1->link;
}
while(temp2!=NULL)
{
newNode(temp2->getCoefficient(), temp2->getPower());
temp2 = temp2->link;
}
}
int main()
{
LinkedList p1;
int polA,A,polB,B, option;
        cout<<"\n\tPOLYNOMIAL USING LINKLIST\n";
        do
        {
        cout<<"\n\t\tPOLYNOMIAL 1 \n";
        cout<<"Enter the highest degree of the poynomial equation 1 : ";
        cin>>polA;
        polA =polA+1;
        cout<<"\nEnter the corresponding constant values :\n";
        for(int i=0;i<polA;i++)
        {
         cout<<"  x^"<<i<<" : ";
         cin>>A;
         p1.newNode(A, i);
        }
LinkedList p2;
        cout<<"\t\tPOLYNOMIAL 2 \n";
        cout<<"Enter the highest degree of the poynomial equation 2 : ";
        cin>>polB;
        polB=polB+1;

        cout<<"Enter the corresponding constant values :\n";
        for(int i=0;i<polB;i++)
        {
         cout<<"  x^"<<i<<" : ";
         cin>>B;
         p2.newNode(B, i);
        }

  cout<<"\n___________________________________________________"//
                 "______________\n\t";
cout<<"\nPolynomial sum: \n\t";
        p1.printPolynomial();
        cout<<"\n\t";
p2.printPolynomial();
cout<<"\n--------------------------------------------------"//
               "-----------------\n\t";
LinkedList sum;

sum.addPolynomial(p1, p2);

cout<<"\n\t";
sum.printPolynomial();
cout<<"\n___________________________________________________"//
               "________________\n";

cout<<"\nDo you want to continue-1 or Exit-0 : ";
cin>>option;
}while(option != 0);
cout<<"\n\tEND\n";
return 0;
}
```

SAMPLE INPUT-OUTPUT



```
        POLYNOMIAL USING LINKLIST

                POLYNOMIAL 1
Enter the highest degree of the poynomial equation 1 : 5

Enter the corresponding constant values :
  x^0 : 2
  x^1 : 3
  x^2 : 4
  x^3 : 0
  x^4 : 1
  x^5 : 4
                POLYNOMIAL 2
Enter the highest degree of the poynomial equation 2 : 4
Enter the corresponding constant values :
  x^0 : 7
  x^1 : 6
  x^2 : 2
  x^3 : 1
  x^4 : 2


__________________________________________________________________

Polynomial sum:
        2x^0 + 3x^1 + 4x^2 + 0x^3 + 1x^4 + 4x^5
        7x^0 + 6x^1 + 2x^2 + 1x^3 + 2x^4
--------------------------------------------------------------------

        9x^0 + 9x^1 + 6x^2 + 1x^3 + 3x^4 + 4x^5
____________________________________________________________________

Do you want to continue-1 or Exit-0 : 1

                POLYNOMIAL 1
Enter the highest degree of the poynomial equation 1 : 3

Enter the corresponding constant values :
  x^0 : 2
  x^1 : 0
  x^2 : 3
  x^3 : -1
                POLYNOMIAL 2
Enter the highest degree of the poynomial equation 2 : 4
Enter the corresponding constant values :
  x^0 : 2
  x^1 : -2
  x^2 : 4
  x^3 : 2
  x^4 : 5


__________________________________________________________________

Polynomial sum:
        2x^0 + 3x^1 + 4x^2 + 0x^3 + 1x^4 + 4x^5 + 2x^0 + 0x^1 + 3x^2 + -1x^3
        2x^0 + -2x^1 + 4x^2 + 2x^3 + 5x^4
--------------------------------------------------------------------

        4x^0 + 1x^1 + 8x^2 + 2x^3 + 6x^4 + 4x^5 + 2x^0 + 0x^1 + 3x^2 + -1x^3
____________________________________________________________________

Do you want to continue-1 or Exit-0 : 0

        END
```

SEARCHING IN LINKED LIST

AIM

To understand the searching of elements in linked list using the function SEARCHITEM().

ALGORITHM

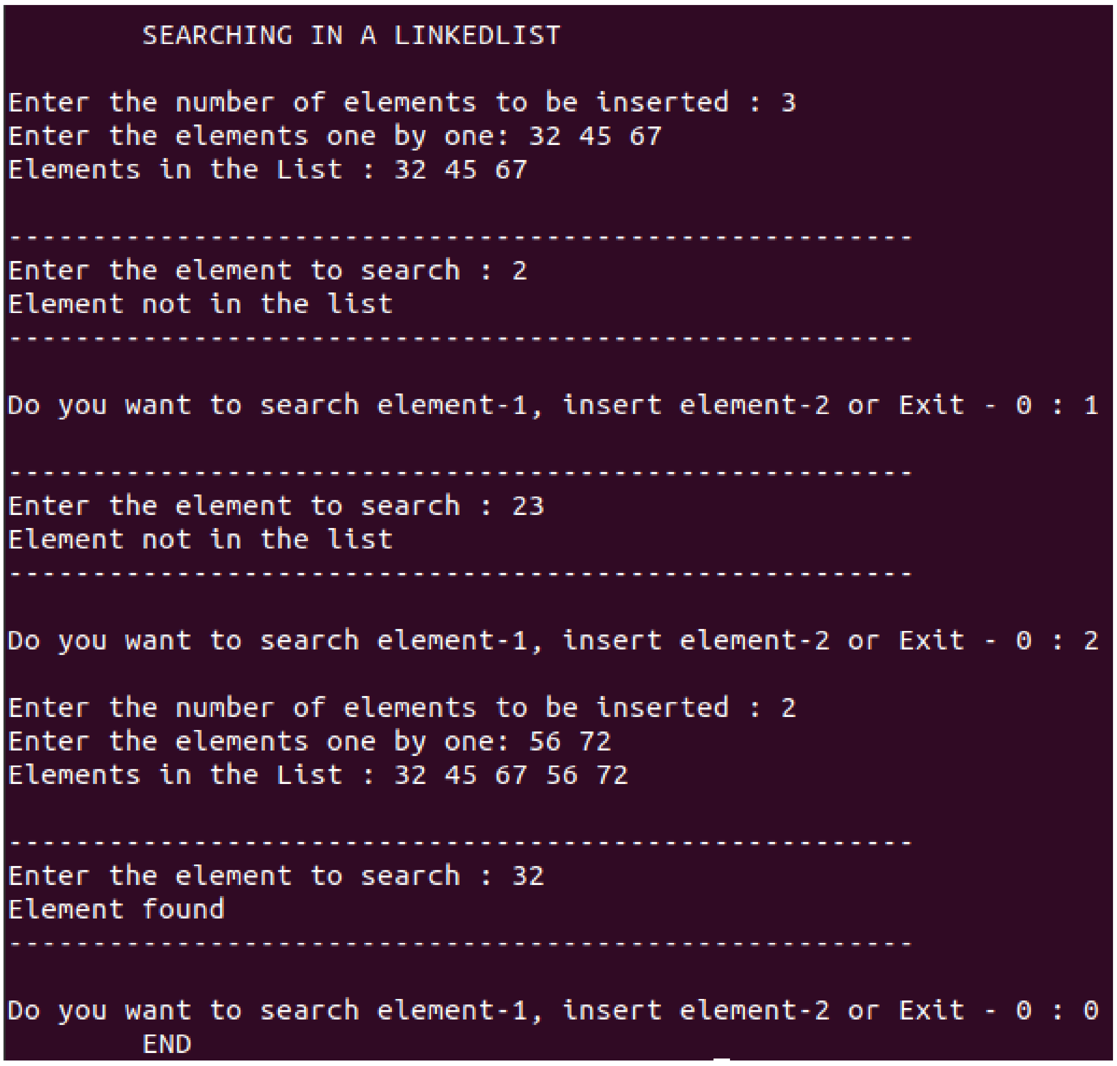
```
    SEARCHITEM( ITEM )
        1.  Create TEMP Node
        2.  Set TEMP = HEAD
        3.  Repeat the step while TEMP!= NULL then
                (i)     if TEMP->INFO = ITEM Then
                            (a) RETURN TEMP
                (ii)    end if condition
                (iv)    Set TEMP = TEMP->LINK
        4.  end while loop
        5.  print :NOT FOUND"
        6.  RETURN NULL
        7.  Exit
```

PROGRAM

```
#include<iostream>
using namespace std;

class LinkedList;

class Node
{
int INFO;
Node* LINK;
    public:
    Node()
    {
    INFO = 0;
    LINK = NULL;
    }
        Node(int INFO)
        {
            this->INFO = INFO;
            this->LINK = NULL;
        }
        friend class LinkedList;
};

class LinkedList
{
Node* head;
    public:
    LinkedList()
    {
    head = NULL;
    }

void addNode(int INFO);
void printNodes();
Node* searchItems(int item);
void deleteNode(int key);
};

Node* LinkedList::searchItems(int item)
{
Node* temp = head;
while (temp !=NULL)
{
if(temp->INFO==item)
{
return temp;
}
temp = temp->LINK;
}
cout<<"Element not in the list"<<endl;
return NULL;
}

void LinkedList::printNodes()
{
  cout<<"Elements in the List : ";

if(head==NULL)
{
cout<<"UNDERFLOW\n";
return;
}
Node* temp = head;

while(temp !=NULL)
{
if(temp->INFO<10)
{
cout<<" ";
}
cout<<temp->INFO<<" ";
temp = temp->LINK;
}
cout<<"\n";
}

void LinkedList::addNode(int INFO)
{
Node* NewNode = new Node(INFO);

if(head == NULL)
{
head = NewNode;
return;
}
Node* temp = head;

while (temp->LINK != NULL)
{
temp = temp->LINK;
}
temp->LINK = NewNode;
}

int main()
{
LinkedList List;

cout<<"\n\tSEARCHING IN A LINKEDLIST\n";
int option,term;

do
```

```
{
int nodeValue;
cout<<"\nEnter the number of elements to be inserted : ";
cin>>term;
cout<<"Enter the elements one by one: ";

for(int i=0;i<term;i++)
{
cin>>nodeValue;
List.addNode(nodeValue);
}

List.printNodes();

do
{
int searchKey;
cout<<"\n------------------------------------------------------";
cout<<"\nEnter the element to search : ";
cin>>searchKey;
Node* address = List.searchItems(searchKey);

if(address)
{
cout<<"Element found "<<endl;
}
cout<<"------------------------------------------------------\n";

cout<<"\nDo you want to search element-1, insert element-2 or Exit - 0 : ";
cin>>option;

}while(option == 1);

}while(option != 0);

cout<<"\tEND"<<endl;
return 0;
}
```

SAMPLE INPUT-OUTPUT



```
        SEARCHING IN A LINKEDLIST

Enter the number of elements to be inserted : 3
Enter the elements one by one: 32 45 67
Elements in the List : 32 45 67

------------------------------------------------------
Enter the element to search : 2
Element not in the list
------------------------------------------------------

Do you want to search element-1, insert element-2 or Exit - 0 : 1

------------------------------------------------------
Enter the element to search : 23
Element not in the list
------------------------------------------------------

Do you want to search element-1, insert element-2 or Exit - 0 : 2

Enter the number of elements to be inserted : 2
Enter the elements one by one: 56 72
Elements in the List : 32 45 67 56 72

------------------------------------------------------
Enter the element to search : 32
Element found
------------------------------------------------------

Do you want to search element-1, insert element-2 or Exit - 0 : 0
        END
```

INSERTION ON SORTED LINKED LIST

AIM

To understand the insertion of elements in the sorted linked list using functions such as INSERT().

ALGORITHM

```
    INSERT( ITEM )
        1.  Create NEW Node
        2.  Set NEW = HEAD
        3.  if ITEM < HEAD->INFO then
                (i)     Create TEMP Node
                (ii)    Set TEMP-> LINK = HEAD
                (iii)   Set HEAD = TEMP
                (iv)    RETURN
        4.  end if condition
        5.  Repeat the steps while NEW->LINK != NULL then
                (i)     Create SAVE with ITEM
                (ii)    Set SAVE = NEW->LINK
                (iii)   if ITEM > NEW->INFO and ITEM < SAVE->INFO
                            (a) Set NEW->LINK = SAVE
                            (b) Set NEW->LINK = TEMP
                            (c) RETURN
                (iv)    Set NEW = NEW->LINK
        6.  end while loop
        7.  if ITEM > NEW->INFO then
                (i)     Create TEMP with ITEM
                (ii)    Set NEW->LINK = TEMP
        8.  end if condition
        9.  Exit
```

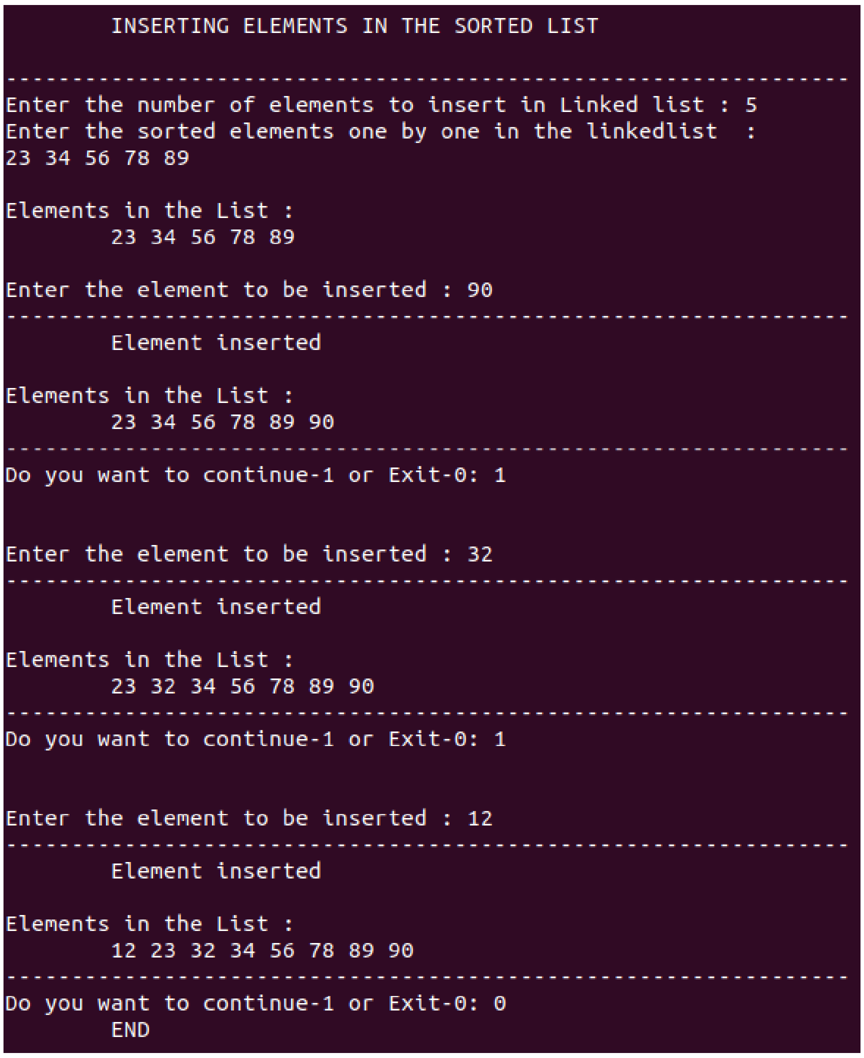
PROGRAM

```
#include <iostream>
using namespace std;

class Node
{
```

```
int DATA;
Node* LINK;
public:
Node()
{
this->DATA = 0;
this->LINK = NULL;
}
Node(int val)
{
this->DATA = val;
this->LINK = NULL;
}
int getData()
{
return DATA;
}
friend class LinkedList;
};

class LinkedList
{
Node* head;
public:
LinkedList()
{
head = NULL;
}
void create(int val);
void print();
void insert(int val);
};

void LinkedList::create(int val)
{
Node* newNode = new Node(val);
if(head==NULL)
{
head = newNode;
return;
}
```

```
Node* temp = head;
while(temp->LINK!=NULL)
{
temp = temp->LINK;
}
temp->LINK = newNode;
}

void LinkedList::print()
{
Node* temp = head;
while(temp!=NULL)
{
cout<<temp->DATA<<" ";
temp = temp->LINK;
}
}

void LinkedList::insert(int val)
{
Node* temp = head;
if(val<head->getData())
{

Node* newNode = new Node(val);
newNode->LINK = head;
head = newNode;
return;
}
while(temp->LINK!=NULL)
{
Node* succ = temp->LINK;
if(val>temp->getData() and val<succ->getData())
{

Node* newNode = new Node(val);
newNode->LINK = succ;
temp->LINK = newNode;
return;
}
temp = temp->LINK;
}
if(val>temp->getData())
{
Node* newNode = new Node(val);
temp->LINK = newNode;
}
}

int main()
{
LinkedList LIST;
cout<<"\n\tINSERTING ELEMENTS IN THE SORTED LIST\n";
int terms,option,val;

cout<<"\n----------------------------------------------------------------";
 cout<<"\nEnter the number of elements to insert in Linked list : ";
 cin>>terms;
 cout<<"Enter the sorted elements one by one in the linkedlist  : ";
 for(int i=0;i<terms;i++)
 {
cin>>val;
LIST.create(val);
}
cout<<"\nElements in the List : \n\t";
LIST.print();
do
{
cout<<"\n\nEnter the element to be inserted : ";
cin>>val;
LIST.insert(val);
cout<<"----------------------------------------------------------------\n";
cout<<"\tElement inserted \n";
cout<<"\nElements in the List : \n\t";
LIST.print();
cout<<"\n----------------------------------------------------------------\n";
cout<<"Do you want to continue-1 or Exit-0: ";
cin>>option;
}while(option != 0);
cout<<"\tEND\n"<<endl;
return 0;
}
```

SAMPLE INPUT-OUTPUT



```
        INSERTING ELEMENTS IN THE SORTED LIST

----------------------------------------------------------------
Enter the number of elements to insert in Linked list : 5
Enter the sorted elements one by one in the linkedlist  :
23 34 56 78 89

Elements in the List :
        23 34 56 78 89

Enter the element to be inserted : 90
----------------------------------------------------------------
        Element inserted

Elements in the List :
        23 34 56 78 89 90
----------------------------------------------------------------
Do you want to continue-1 or Exit-0: 1


Enter the element to be inserted : 32
----------------------------------------------------------------
        Element inserted

Elements in the List :
        23 32 34 56 78 89 90
----------------------------------------------------------------
Do you want to continue-1 or Exit-0: 1


Enter the element to be inserted : 12
----------------------------------------------------------------
        Element inserted

Elements in the List :
        12 23 32 34 56 78 89 90
----------------------------------------------------------------
Do you want to continue-1 or Exit-0: 0
        END
```

MERGING TWO SORTED LIST

AIM

To understand the MERGING of two different Sorted linked list into a single sorted linked list.

ALGORITHM

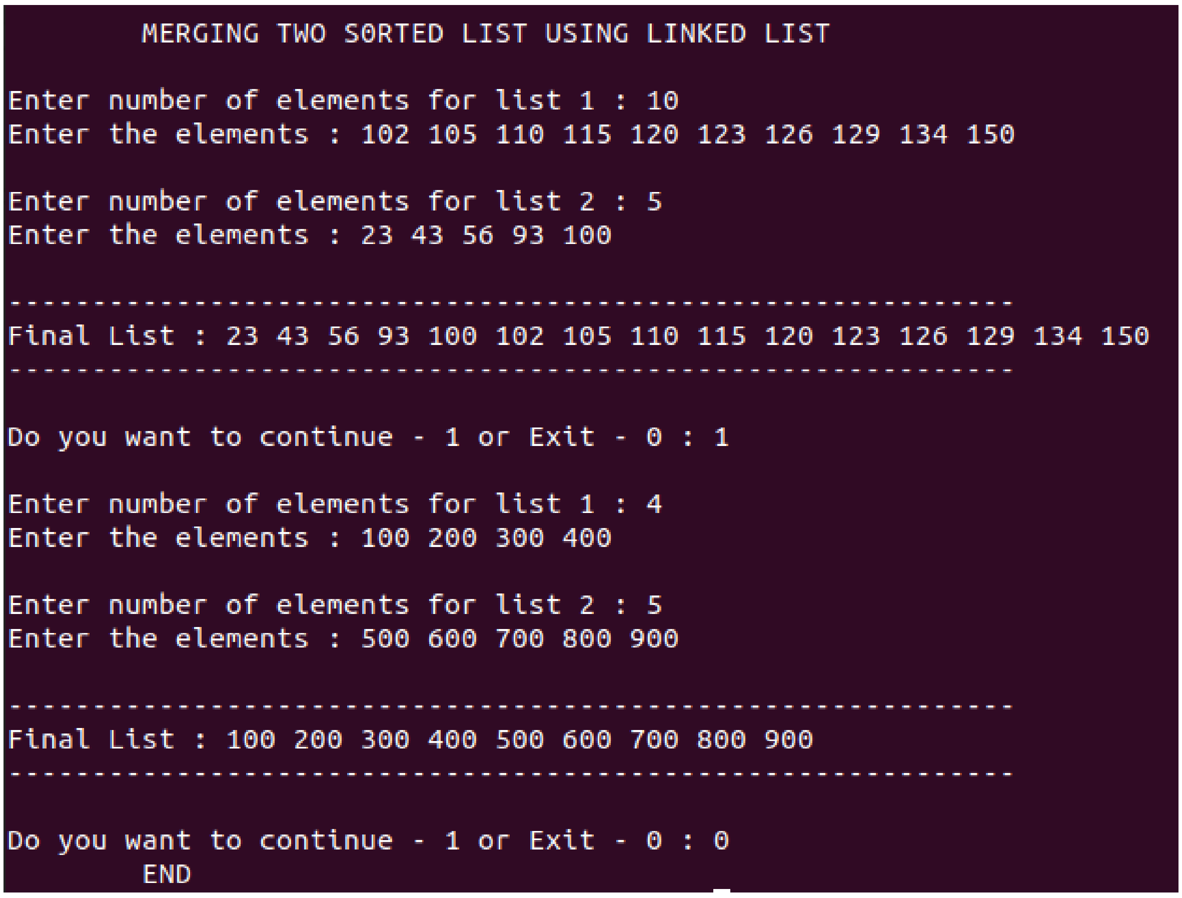
```
MERGE( ARRAY1 , TEMPA , ARRAY2 , TEMPB )
    1.  Set NA = 1
    2.  Set NB = 1
    3.  Set PTR = 1
    4.  Create ARRAY3
    5.  Repeat the steps while NA <= TEMPA and NB <= TEMPB
            (i)     if ARRAY1[NA] = ARRAY2[NB]
                        (a) Set ARRAY3[PTR] = ARRAY1[NA]
                        (b) Set NA = NA+1
                        (c) Set PTR = PTR+1
            (ii)    else
                        (a) ARRAY3[PTR] = ARRAY2[NB]
                        (b) Set NB = NB+1
                        (c) Set PTR = PTR+1
            (iv)    end if condition
    6.  end while loop
    7.  if NA > TEMPA then
            (i)     Repeat the steps for K=0 to K < (ARRAY2-NB)
                        (a) ARRAY3[PTR + K] = B[NB + K]
            (ii)    end while loop
    8.  else
            (i)     Repeat the steps while K=0 to K < (ARRAY1-NA)
                        (a) ARRAY3[PTR +K] = A[NA+K]
            (ii)    end while loop
    9.  end if condition
    10. Exit
```

PROGRAM

```
#include<iostream>
using namespace std;

class node
```

```
{
    public:
        int data;
        node* next;
};

void insert(node** head_ref, int new_data)
{
    node* temp = new node();
    node* temp1 = *head_ref;
    temp->data = new_data;
    temp->next = NULL;
    if (*head_ref == NULL)
    {
        *head_ref = temp;
        return;
    }
    else
    {
        while (temp1->next != NULL)
        {
            temp1 = temp1->next;
        }
        temp1->next = temp;
        return;
    }
}

node* Sorted(node* A, node* B)
{
    node* sorted_list = NULL;
    if (A == NULL)
    {
        return B;
    }
    else if (B == NULL)
    {
        return A;
    }
    if (A->data > B->data)
    {
        sorted_list = B;
        sorted_list->next = Sorted(A, B->next);
    }
    else
    {
        sorted_list = A;
        sorted_list->next = Sorted(A->next, B);
    }
    return sorted_list;
}

void mergesplit(node* start, node** A, node** B)
{
    node* fast;
    node* slow;
    fast = start->next;
    slow = start;
    while (fast != NULL)
    {
        fast = fast->next;
        if (fast != NULL)
        {
            slow = slow->next;
            fast = fast->next;
        }
    }
    *A = start;
    *B = slow->next;
    slow->next = NULL;
}

void MergeSort(node** head)
{
    node* start = *head;
    node* A;
    node* B;
    if (start == NULL || start->next == NULL)
    {
        return;
    }
    mergesplit(start, &A, &B);
    MergeSort(&A);
    MergeSort(&B);
    *head = Sorted(A, B);
}

void display(node* node)
{
    while (node != NULL)
    {
        cout << node->data << " ";
        node = node->next;
    }
}

int main()
{
    int ch;
    cout << "\n\tMERGING TWO SORTED LIST USING LINKED LIST" << endl;
    do
    {
        node* head = NULL;
        for (int i = 1; i < 3; i++)
        {
        cout << "\nEnter number of elements for list " << i << " : ";
        int num, value;
        cin >> num;
        cout << "Enter the elements : ";
        for (int j = 0; j < num; j++)
        {
            cin >> value;
            insert(&head, value);
        }
        MergeSort(&head);
        }
        cout<<"\n------------------------------------------------------------";
        cout << "\nFinal List : ";
        display(head);
        cout<<"\n------------------------------------------------------------\n";
        cout << "\nDo you want to continue - 1 or Exit - 0 : ";
        cin >> ch;
    } while (ch != 0);
    cout<<"\tEND\n";
    return 0;
}
```

SAMPLE INPUT-OUTPUT



```
        MERGING TWO S0RTED LIST USING LINKED LIST

Enter number of elements for list 1 : 10
Enter the elements : 102 105 110 115 120 123 126 129 134 150

Enter number of elements for list 2 : 5
Enter the elements : 23 43 56 93 100

------------------------------------------------------------
Final List : 23 43 56 93 100 102 105 110 115 120 123 126 129 134 150
------------------------------------------------------------

Do you want to continue - 1 or Exit - 0 : 1

Enter number of elements for list 1 : 4
Enter the elements : 100 200 300 400

Enter number of elements for list 2 : 5
Enter the elements : 500 600 700 800 900

------------------------------------------------------------
Final List : 100 200 300 400 500 600 700 800 900
------------------------------------------------------------

Do you want to continue - 1 or Exit - 0 : 0
        END
```

DELETION FROM LINKED LIST

AIM

To understand the Deletion in the linked list using the function DELETE()

ALGORITHM

```
DELTETE( ITEM )
    1.  if HEAD->INFO == ITEM
            (i)     Set HEAD = HEAD->LINK
            (ii)    RETURN
    2.  end if condition
    3.  Create PRE_NODE
    4.  Set PRE_NODE = HEAD
    5.  Create CURRENT
    6.  Set PTR = HEAD->LINK
    7.  Repeat the steps while PTR != NULL THEN
            (i)     if PTR->INFO = ITEM
                        (a) Set PRE_NODE->LINK = PTR->LINK
                        (b) DELETE PTR
                        (a) RETURN 0
            (ii)    end if condition
            (iii)   Set PRE_NODE = PTR
            (iV)    Set PTR = PTR->LINK
    8.  end while loop
    7.  print "NOT FOUND"
    8.  RETURN -1
    9.  Exit
```

PROGRAM

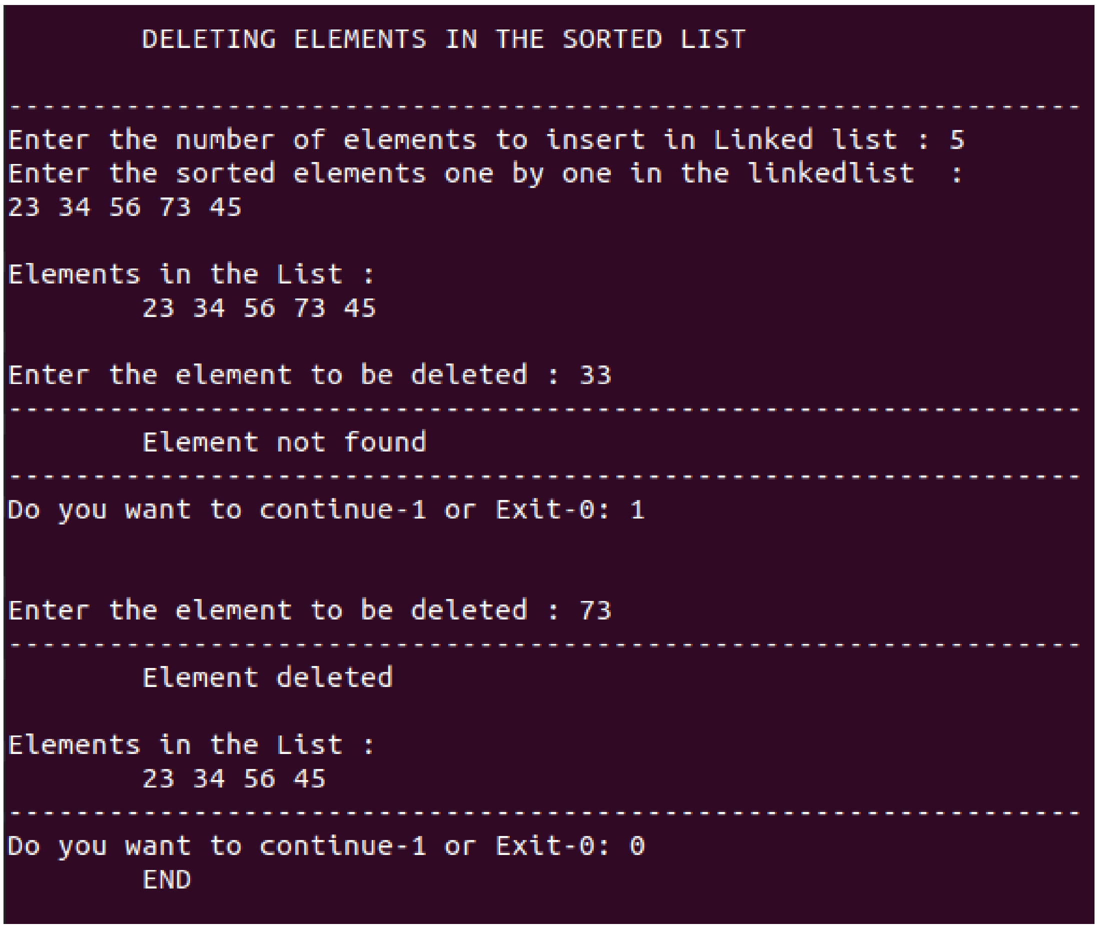
```
#include <iostream>
using namespace std;
class LinkedList;
class Node
{
int INFO;
Node* LINK;
public:
Node()
{
```

```
INFO = 0;
LINK = NULL;
}
    Node(int INFO){
        this->INFO = INFO;
        this->LINK = NULL;
    }
    friend class LinkedList;
};
class LinkedList
{
Node* head;
    public:
LinkedList()
{
head = NULL;
}
void add(int INFO);
void print();
Node* searchItems(int item);
int deleteNode(int key);
};
Node* LinkedList::searchItems(int item)
{
Node* temp = head;
while (temp !=NULL)
{
if(temp->INFO==item)
{
return temp;
}
temp = temp->LINK;
}
cout<<"Element not in the list"<<endl;
return NULL;
}
void LinkedList::print()
{
if(head==NULL)
{
cout<<"UNDERFLOW"<<endl;
```

```
return;
}
Node* temp = head;
while(temp !=NULL)
{
if(temp->INFO<10)
{
cout<<" ";
}
cout<<temp->INFO<<" ";
temp = temp->LINK;
}
}
void LinkedList::add(int INFO)
{
Node* NewNode = new Node(INFO);
if(head == NULL){
head = NewNode;
return;
}
Node* temp = head;
while (temp->LINK != NULL)
{
temp = temp->LINK;
}
temp->LINK = NewNode;
}
int LinkedList::deleteNode(int key)
{
if(head->INFO==key)
{
head = head->LINK;
return 0;
}
Node* prevNode = head;
Node* current = head->LINK;
while(current!=NULL)
{
if(current->INFO==key)
{
prevNode->LINK = current->LINK;
delete current;
return 0;
}
prevNode = current;
current = current->LINK;
}
cout<<"\tElement not found "<<endl;
return -1;
}
int main()
{
LinkedList LIST;
cout<<"\n\tINSERTING ELEMENTS IN THE SORTED LIST\n";
int terms,option,val, del;

cout<<"\n----------------------------------------------------------------";
 cout<<"\nEnter the number of elements to insert in Linked list : ";
 cin>>terms;
 cout<<"Enter the sorted elements one by one in the linkedlist  : ";
 for(int i=0;i<terms;i++)
 {
cin>>val;
LIST.add(val);
}
cout<<"\nElements in the List : \n\t";
LIST.print();
do
{
cout<<"\n\nEnter the element to be deleted : ";
cin>>val;
cout<<"---------------------------------------------------------------\n";
del = LIST.deleteNode(val);
if(del != (-1))
{
 cout<<"\tElement deleted \n";
 cout<<"\nElements in the List : \n\t";
 LIST.print();
 cout<<"\n";
}
cout<<"---------------------------------------------------------------\n";
cout<<"Do you want to continue-1 or Exit-0: ";
```

```
cin>>option;

}while(option != 0);
cout<<"\tEND\n"<<endl;
return 0;
}
```

SAMPLE INPUT-OUTPUT



```
        DELETING ELEMENTS IN THE SORTED LIST

----------------------------------------------------------------
Enter the number of elements to insert in Linked list : 5
Enter the sorted elements one by one in the linkedlist  :
23 34 56 73 45

Elements in the List :
        23 34 56 73 45

Enter the element to be deleted : 33
----------------------------------------------------------------
        Element not found
----------------------------------------------------------------
Do you want to continue-1 or Exit-0: 1


Enter the element to be deleted : 73
----------------------------------------------------------------
        Element deleted

Elements in the List :
        23 34 56 45
----------------------------------------------------------------
Do you want to continue-1 or Exit-0: 0
        END
```

OPERATIONS ON CIRCULAR HEADER LIST

AIM

To understand the working of the Circular Header List which include INSERT() AND DELETE() using the Linked List.

ALGORITHM

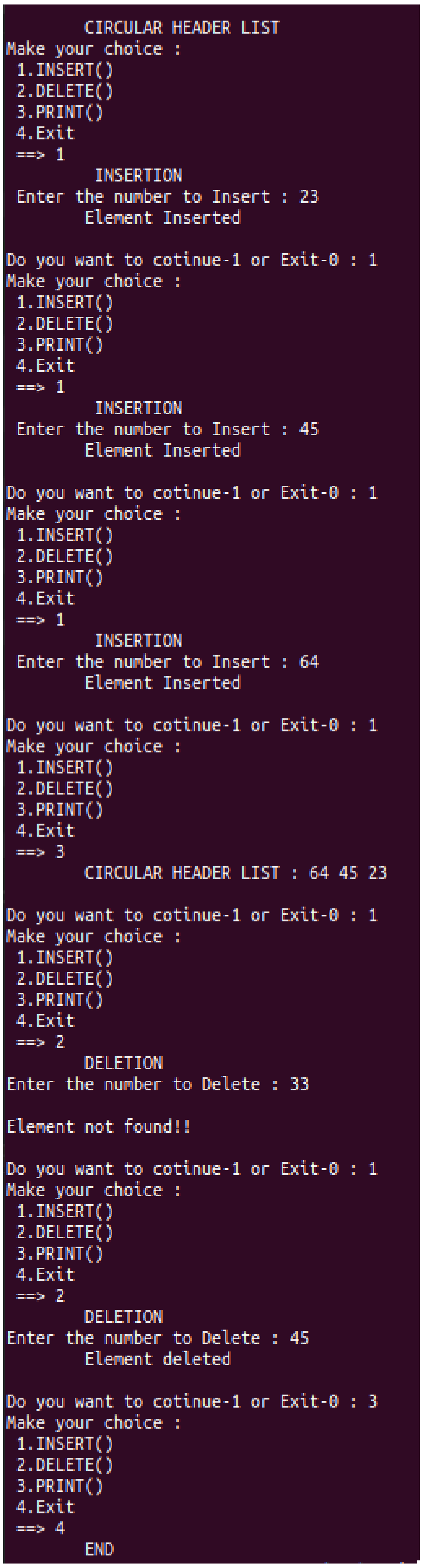
```
INSERT( NODE , ITEM )
    1.  Create NEW node
    2.  Set NEW->INFO = ITEM
    3.  Set NEW->LINK = NODE
    4.  if NODE != NULL then
            (i)     Create TEMP node
            (ii)    Set TEMP = NODE
            (iii)   Repeat the steps while TEMP->LINK != NODE then
                    (a) Set TEMP = TEMP->LINK
            (iv)    end while loop
            (v)     Set TEMP->LINK = NEW
    5.  else
            (i)     Set NEW->LINK = NEW
    6.  end if condition
    7.  print "INSERTED"
    8.  Set NODE = NEW
    9.  Exit

DELETE( NODE , ITEM )
    1.  if HEAD == NULL then
            (i) RETURN
    2.  End if condition
    3.  if HEAD->INFO = ITEM and HEAD->LINK = HEAD then
            (i)     Set HEAD = NULL
            (ii)    RETURN
    4.  End if condition
    5.  Create LAST Node and SUB Node
    6.  Set LAST = HEAD
    7.  if HEAD->INFO == ITEM then
            (i)     while LAST->INFO != HEAD
                        (a) Set LAST = LAST->LINK
            (ii)    End while loop
            (iii)   Set LAST->LINK = HEAD->LINK
```

```
                (iv)    Set HEAD = LAST->LINK
                (v)     RETURN
        8.  End if condition
        9.  Repeat the steps while LAST->LINK != HEAD and
                    (LAST->LINK)->INFO != ITEM then
                (i)     Set LAST = LAST->LINK
        10. End while loop
        11. if (LAST->LINK)->INFO == ITEM then
                (i)     Set SUB = LAST->LINK
                (ii)    LAST->LINK = SUB->LINK
                (iii)   print "ELEMENT DELETED"
        12. else
                (i)     print "NOT FOUND"
        13. End if condition
        14. Exit
```

PROGRAM

```
#include <iostream>
using namespace std;
class Node
{
    public:
    int data;
    Node* next;
};

void insert(Node** head_ref, int data)
{
    Node* ptr1 = new Node();
    ptr1->data = data;
    ptr1->next = *head_ref;

    if (*head_ref != NULL)
    {
        Node* temp = *head_ref;
        while (temp->next != *head_ref)
             temp = temp->next;
        temp->next = ptr1;
    }
    else
    {
            ptr1->next = ptr1;
    }
    cout<<"\tElement Inserted\n";
    *head_ref = ptr1;
}

void printList(Node* head)
{
    Node* temp = head;

    if (head != NULL)
    {
    do
    {
        cout << temp->data << " ";
        temp = temp->next;
    } while (temp != head);
    }
    cout << endl;
}

void deleteNode(Node** head, int key)
{
    if (*head == NULL)
    {
        return;
    }
    if ((*head)->data == key && (*head)->next == *head)
    {
        *head = NULL;
        return;
    }

    Node* last = *head, * d;

    if ((*head)->data == key)
    {
        while (last->next != *head)
        {
            last = last->next;
        }
    last->next = (*head)->next;
    *head = last->next;
    return;
    }
    while (last->next != *head && last->next->data != key)
    {
        last = last->next;
    }
    if (last->next->data == key)
    {
        d = last->next;
        last->next = d->next;
        cout<<"\tElement deleted\n";
    }
    else
    {
        cout << "\nElement not found!!" << endl;
    }
}

int main()
{
    Node* head = NULL;
    int ch, option;
    cout << "\n\tCIRCULAR HEADER LIST\n" ;

    do
    {
    cout << "Make your choice : \n 1.INSERT()\n 2.DELETE()\n 3.PRINT()"//
        "\n 4.Exit \n ==> ";
    cin >> ch;

    switch (ch)
    {

    case 1:
        int val;
        cout << "\t INSERTION \n Enter the number to Insert : ";
        cin >> val;
        insert(&head, val);
        break;

    case 2:
        cout << "\tDELETION \nEnter the number to Delete : ";
        cin >> val;
        deleteNode(&head, val);
        break;

    case 3:
        cout << "\tCIRCULAR HEADER LIST : ";
        printList(head);
        break;

    case 4:
        break;

    default:
        cout << "Wrong choice!!" << endl;
        break;
    }

    if (ch !=4 )
    {
        cout<<"\nDo you want to continue-1 or Exit-0 : ";
        cin>>option;

        if (option == 0)
        {
            ch=4;
        }
    }
    }while(ch!=4);

    cout<<"\tEND\n";
    return 0;
}
```

SAMPLE INPUT-OUTPUT



```
        CIRCULAR HEADER LIST
Make your choice :
 1.INSERT()
 2.DELETE()
 3.PRINT()
 4.Exit
 ==> 1
         INSERTION
 Enter the number to Insert : 23
        Element Inserted

Do you want to cotinue-1 or Exit-0 : 1
Make your choice :
 1.INSERT()
 2.DELETE()
 3.PRINT()
 4.Exit
 ==> 1
         INSERTION
 Enter the number to Insert : 45
        Element Inserted

Do you want to cotinue-1 or Exit-0 : 1
Make your choice :
 1.INSERT()
 2.DELETE()
 3.PRINT()
 4.Exit
 ==> 1
         INSERTION
 Enter the number to Insert : 64
        Element Inserted

Do you want to cotinue-1 or Exit-0 : 1
Make your choice :
 1.INSERT()
 2.DELETE()
 3.PRINT()
 4.Exit
 ==> 3
        CIRCULAR HEADER LIST : 64 45 23

Do you want to cotinue-1 or Exit-0 : 1
Make your choice :
 1.INSERT()
 2.DELETE()
 3.PRINT()
 4.Exit
 ==> 2
        DELETION
Enter the number to Delete : 33

Element not found!!

Do you want to cotinue-1 or Exit-0 : 1
Make your choice :
 1.INSERT()
 2.DELETE()
 3.PRINT()
 4.Exit
 ==> 2
        DELETION
Enter the number to Delete : 45
        Element deleted

Do you want to cotinue-1 or Exit-0 : 3
Make your choice :
 1.INSERT()
 2.DELETE()
 3.PRINT()
 4.Exit
 ==> 4
        END
```

SEARCHING IN TWO-WAY LINKED LIST

AIM

To understand the working of the Two-Way linked list searching algorithm using SEARCH().

ALGORITHM

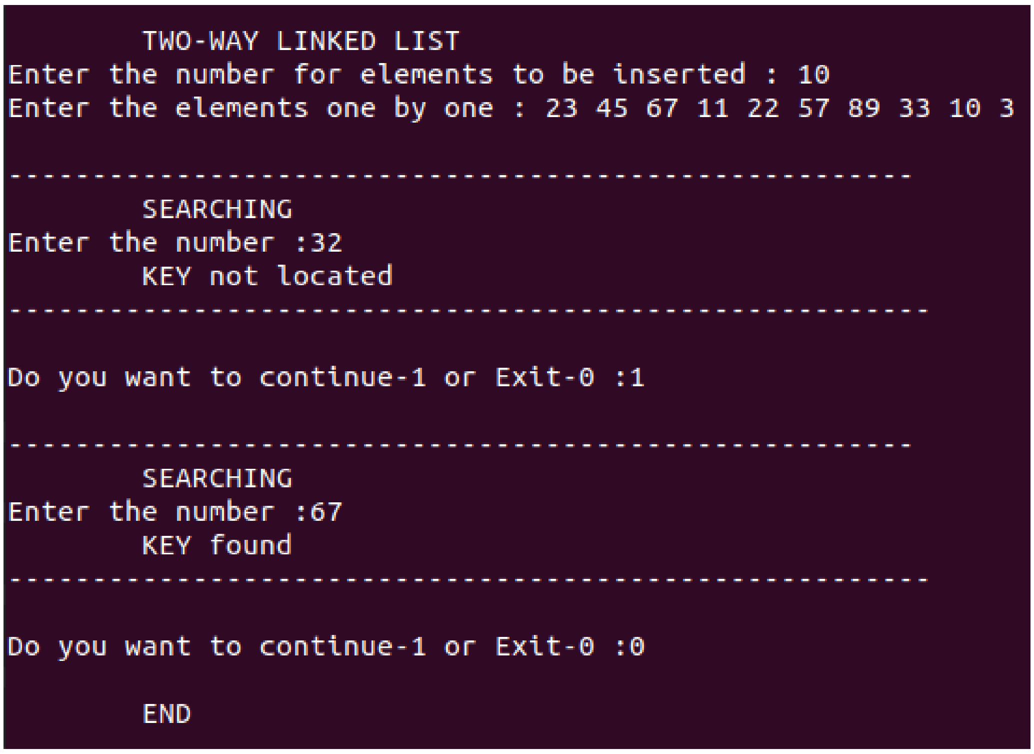
```
SEARCH( NODE , ITEM )
    1.  if NODE == NULL Then
            (i)     RETURN
    2.  End if condition
    3.  Create NEW node
    4.  NODE =  NODE
    5.  Repeat the the steps while NEW != NULL
            (i)     if NEW->INFO = ITEM then
                        (a) print ITEM found
                        (b) RETURN
            (ii)   End if condition
    6.  Set NEW = NEW->LINK
    7.  Print element not found
    8.  Exit
```

PROGRAM

```
 #include <iostream>
using namespace std;

class Node {
public:
int data;
Node* next;
Node* prev;
};

void insert(Node** head_ref, int new_data)
{
Node* new_node = new Node();
new_node->data = new_data;

new_node->next = (*head_ref);
new_node->prev = NULL;
```

```
if ((*head_ref) != NULL)
(*head_ref)->prev = new_node;

(*head_ref) = new_node;

}



void search(Node** head_ref, int key){
    if ((*head_ref) == NULL)
        return;
    Node* current = *head_ref;
    while(current != NULL){
        if(current->data==key){
            cout<<"\tKEY found"<<endl;
            return;
        }
        current = current->next;
    }
    cout<<"\tKEY not located"<<endl;
}



int main(){
Node* head = NULL;
cout<<"\n\tTWO-WAY LINKED LIST\n";

int option, terms , val;
cout<<"Enter the number for elements to be inserted : ";
cin>>terms;
cout<<"Enter the elements one by one : ";
for(int i=1 ; i <= terms; i++)
{
 cin>>val;
 insert(&head,val);
}
do
{
cout<<"\n---------------------------------------------------";
cout<<"\n\tSEARCHING\nEnter the number :";
cin>>val;
search(&head,val);
cout<<"-------------------------------------------------------\n";

cout<<"\nDo you want to continue-1 or Exit-0 :";
cin>>option;

}while(option != 0);

cout<<"\n\tEND\n"<<endl;
return 0;
}
```

SAMPLE INPUT-OUTPUT



```
        TWO-WAY LINKED LIST
Enter the number for elements to be inserted : 10
Enter the elements one by one : 23 45 67 11 22 57 89 33 10 3

-------------------------------------------------------
        SEARCHING
Enter the number :32
        KEY not located
-------------------------------------------------------

Do you want to continue-1 or Exit-0 :1

-------------------------------------------------------
        SEARCHING
Enter the number :67
        KEY found
-------------------------------------------------------

Do you want to continue-1 or Exit-0 :0

        END
```

OPERATIONS ON TWO-WAY LINKED LIST

AIM

To understand the operation that include INSERT() and DELETE() in the Two-Way linked list

ALGORITHM

```
INSERT( NODE , ITEM )
    1.  Create NEW node
    2.  Set NEW->INFO = ITEM
    3.  Set NEW->FORW = NODE
    4.  Set NEW->BACK = NULL
    5.  if NODE != NULL then
            (i)     Set NODE->BACK = NEW
    6.  End if condition
    7.  Set NODE = NEW
    8.  Exit

INSERTION (START , ITEM LOC)
    1.  Create NEW node
    2.  Set NEW->INFO = ITEM
    3.  Set NEW->FORW = LOC->FORW
    4.  Set NEW->BACK = LOC
    5.  Set (LOC->FORW)->BACK = NEW
    6.  Set LOC->FORW = NEW
    7.  Exit

DELETE( NODE , ITEM )
    1.  if NODE == NULL then
            (i) RETURN
    2.  End if
    3.  if NODE->INFO == ITEM then
            (i) Set NODE = NODE->FORW
    4.  End if
    5.  if NODE->FORW != NULL then
            (i)     Set (NODE->BACK)->FORW = NODE->FORW
            (ii)    Set (NODE->FORW)->BACK = NODE->BACK
    6.  End if condition
    7.  RETURN
    8.  Exit
```

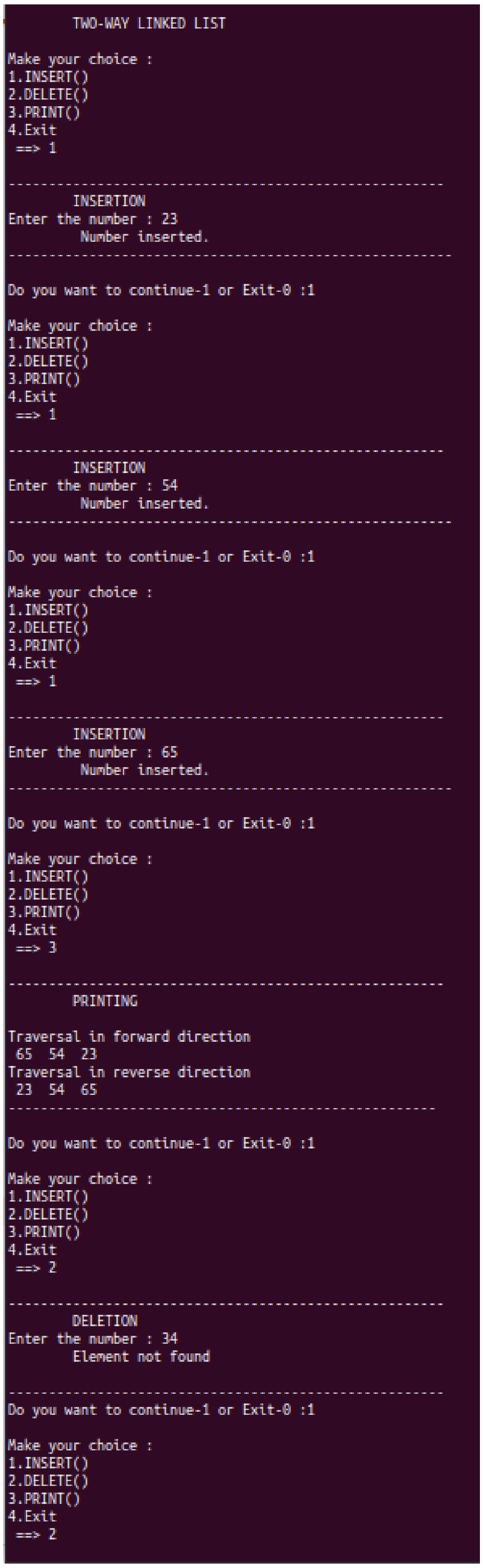
PROGRAM

```
 #include <iostream>
using namespace std;

class Node {
public:
int data;
Node* next;
Node* prev;
};

void insert(Node** head_ref, int new_data)
{
Node* new_node = new Node();
new_node->data = new_data;

new_node->next = (*head_ref);
new_node->prev = NULL;

if ((*head_ref) != NULL)
(*head_ref)->prev = new_node;

(*head_ref) = new_node;
cout<<"\t Number inserted.\n";
}

void printList(Node* node)
{
Node* last;
cout << "\nTraversal in forward direction \n";
while (node != NULL)
{
cout << " " << node->data << " ";
last = node;
node = node->next;
}

cout << "\nTraversal in reverse direction \n";
while (last != NULL)
```

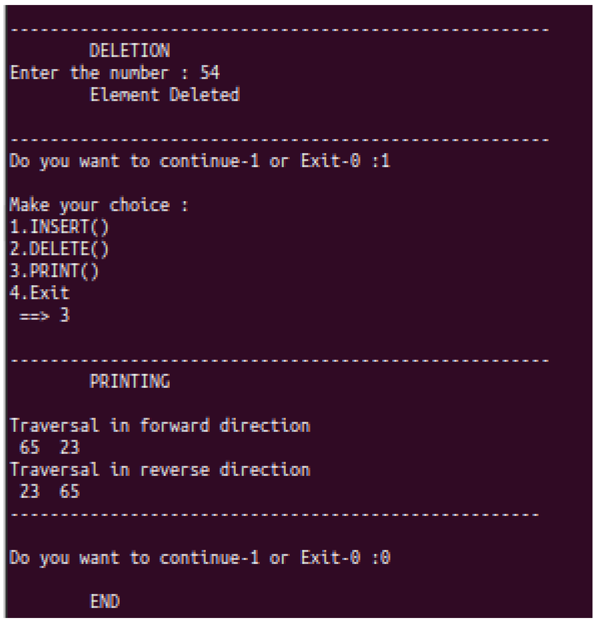
```
{
cout << " " << last->data << " ";
last = last->prev;
}
cout<<endl;
}

void deleteNode(Node** head_ref, Node* del)
{
    if (*head_ref == NULL || del == NULL)
        return;

    if (*head_ref == del)
        *head_ref = del->next;

    if (del->next != NULL)
        del->next->prev = del->prev;

    if (del->prev != NULL)
        del->prev->next = del->next;

    return;
}

void deleteAllOccurOfX(Node** head_ref, int x)
{
   int count = 0;
    if ((*head_ref) == NULL)
        return;

    Node* current = *head_ref;
    Node* next;

    while (current != NULL)
    {
        if (current->data == x)
        {
            next = current->next;
            deleteNode(head_ref, current);
            current = next;
            cout<<"\tElement Deleted\n";
            count =1;
        }
        else
            current = current->next;
    }
    if (current == NULL && count == 0)
     cout<<"\tElement not found\n";
}

void search(Node** head_ref, int key)
{
    if ((*head_ref) == NULL)
        return;
    Node* current = *head_ref;
    while(current != NULL){
        if(current->data==key)
        {
            cout<<"\nKEY found"<<endl;
            return;
        }
        current = current->next;
    }
    cout<<"\nKEY not located"<<endl;
}

int main()
{
Node* head = NULL;
cout<<"\n\tTWO-WAY LINKED LIST\n";

int mainOption;
do
{
cout<<"\nMake your choice : \n1.INSERT()\n2.DELETE()\n3.PRINT()""//
        "\n4.Exit\n ==> ";
cin>>mainOption;
switch (mainOption)
{

case 1:
cout<<"\n------------------------------------------------------";
```

```
int val;
cout<<"\n\tINSERTION\n";
cout<<"Enter the number : ";
cin>>val;
insert(&head,val);
cout<<"-------------------------------------------------------\n";
break;

case 2:
cout<<"\n------------------------------------------------------";
cout<<"\n\tDELETION\nEnter the number : ";
cin>>val;
deleteAllOccurOfX(&head, val);
cout<<"\n------------------------------------------------------";
break;

case 3:
cout<<"\n------------------------------------------------------";
cout<<"\n\tPRINTING\n";
printList(head);
cout<<"----------------------------------------------------- \n";
break;
case 4:
   mainOption = 0;
break;
default:
cout<<"\n------------------------------------------------------";
cout<<"\nWrong choice"<<endl;
cout<<"-------------------------------------------------------\n";
break;
}

if (mainOption != 4)
{
cout<<"\nDo you want to continue-1 or Exit-0 :";
cin>>mainOption;
}
}while(mainOption != 0);
cout<<"\n\tEND\n"<<endl;
return 0;
}
```

SAMPLE INPUT-OUTPUT



```
        TWO-WAY LINKED LIST

Make your choice :
1.INSERT()
2.DELETE()
3.PRINT()
4.Exit
 ==> 1

------------------------------------------------------
        INSERTION
Enter the number : 23
         Number inserted.
-------------------------------------------------------

Do you want to continue-1 or Exit-0 :1

Make your choice :
1.INSERT()
2.DELETE()
3.PRINT()
4.Exit
 ==> 1

------------------------------------------------------
        INSERTION
Enter the number : 54
         Number inserted.
-------------------------------------------------------

Do you want to continue-1 or Exit-0 :1

Make your choice :
1.INSERT()
2.DELETE()
3.PRINT()
4.Exit
 ==> 1

------------------------------------------------------
        INSERTION
Enter the number : 65
         Number inserted.
-------------------------------------------------------

Do you want to continue-1 or Exit-0 :1

Make your choice :
1.INSERT()
2.DELETE()
3.PRINT()
4.Exit
 ==> 3

------------------------------------------------------
        PRINTING

Traversal in forward direction
 65  54  23
Traversal in reverse direction
 23  54  65
-----------------------------------------------------

Do you want to continue-1 or Exit-0 :1

Make your choice :
1.INSERT()
2.DELETE()
3.PRINT()
4.Exit
 ==> 2

------------------------------------------------------
        DELETION
Enter the number : 34
        Element not found

------------------------------------------------------
Do you want to continue-1 or Exit-0 :1

Make your choice :
1.INSERT()
2.DELETE()
3.PRINT()
4.Exit
 ==> 2
```



```
-----------------------------------------------------
        DELETION
Enter the number : 54
        Element Deleted

-----------------------------------------------------
Do you want to continue-1 or Exit-0 :1

Make your choice :
1.INSERT()
2.DELETE()
3.PRINT()
4.Exit
 ==> 3

-----------------------------------------------------
        PRINTING

Traversal in forward direction
 65  23
Traversal in reverse direction
 23  65
----------------------------------------------------

Do you want to continue-1 or Exit-0 :0

        END
```

OPERATIONS ON STACK USING ARRAY

AIM

To understand the PUSH() and POP() function using the STACK Data Structure constructed using the Array.

ALGORITHM

```
    PUSH ( ITEM )
        1.  if TOP = MAXSTK then
                (i)     print "OVERFLOW"
                (ii)    RETURN 1;
        2.  End if condition
        3.  Set TOP = TOP+1
        4.  STACK[TOP] = ITEM
        5.  RETURN 0
        6.  Exit

    POP ( ITEM )
        1.  if TOP == -1 then
                (i)     print "UNDERFLOW"
                (ii)    RETURN -1
        2.  End if
        3.  Set ITEM = STACK[TOP]
        4.  Set TOP = TOP-1
        5.  RETURN ITEM
        6.  Exit
```

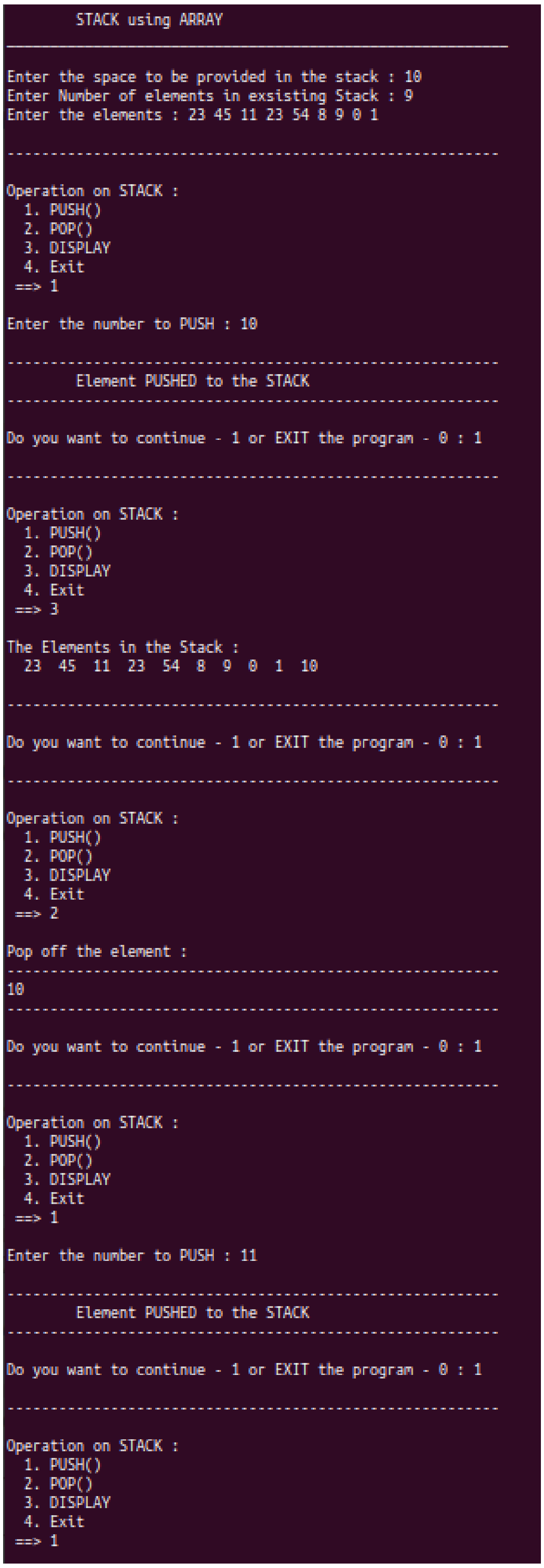
PROGRAM

```
#include<iostream>
using namespace std;

class STACKARRAY
{

 int *Stack;
 int TOP;
 int MAXSTK;
public:
 STACKARRAY( int maxstk)
```

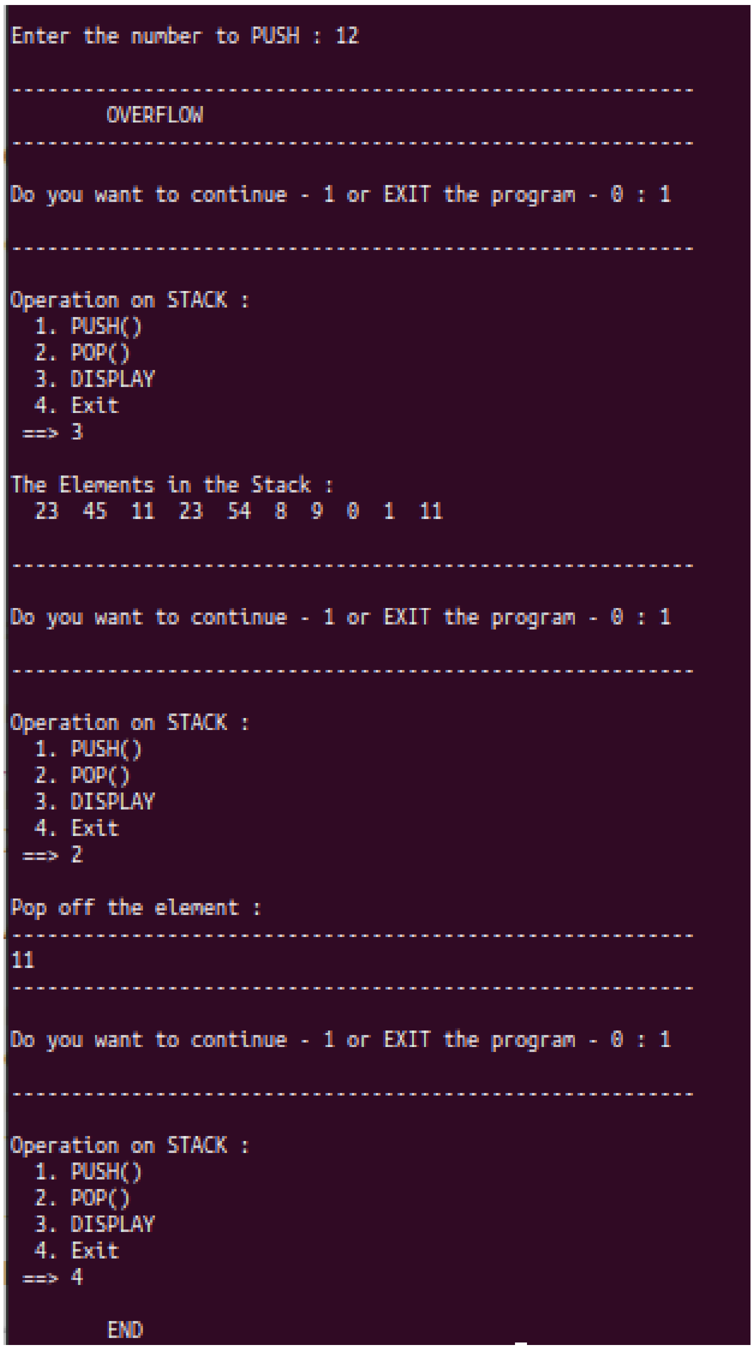
```
{
 TOP  = -1;
 MAXSTK = maxstk;
 Stack = new int[maxstk];
}
int PUSH( int ITEM )
{
 if (TOP == MAXSTK)
 {
   cout<<"\n---------------------------------------------"//
                    "------------\n";
  cout<<"\tOVERFLOW ";
  return 1;
 }
 TOP = TOP+1;
 Stack[TOP] = ITEM;
 return 0;
}
int POP( )
{
 int ITEM;

 if (TOP == -1)
 {
  cout<<"\n---------------------------------------------"//
                    "------------\n";
  cout<<"\tUNDERFLOW ";
  return -1;
 }
 ITEM = Stack[TOP];
  TOP = TOP-1;
 return ITEM;
}
void DISPLAY()
{
 for(int i=0;i<=TOP;i++)
 {
  cout<<"  "<<Stack[i];
 }
 cout<<endl;
}
```

```
};

int main()
{
 int MAXSTACK,terms,element,option,key;
          cout<<"\n\tSTACK using ARRAY \n";
          cout<<"____________________________________________"//
                    "_____________\n";
          cout<<"\nEnter the space to be provided in the stack : ";
          cin>> MAXSTACK;

          STACKARRAY SA(MAXSTACK-1)  ;

          cout<<"Enter Number of elements in exsisting Stack : ";
          cin >> terms;
          cout<<"Enter the elements : ";
          for(int i=0;i<terms;i++)
          {
            cin>>element;
            SA.PUSH(element);
          }
          do
          {     cout<<"\n-----------------------------------------"//
                            "----------------\n";
  cout<<"\nOperation on STACK :\n  1. PUSH()\n  2. POP() \n  3. DISPLAY"//
                      "\n  4. Exit \n ==> ";
  cin>>option;
switch(option)
  {
    case 1:
        cout<<"\nEnter the number to PUSH : ";
        cin>>element;

        key = SA.PUSH(element);
                       if(key != 1)
                       {
                         cout<<"\n-----------------------------------"//
                                "-----------------------\n";
                         cout<<"\tElement PUSHED to the STACK";
                       }
        break;
    case 2:
        cout<<"\nPop off the element : ";
        element = SA.POP();

        if (element != (-1))
        {
                        cout<<"\n-------------------------------------------"//
                                "--------------\n";
              cout<<element;
        }
              break;

                case 3:
                      cout<<"\nThe Elements in the Stack : \n";

                      SA.DISPLAY();

                      break;
                case 4:
                      option = 0;
         break;

    default : cout<<"\nWrong Choice\n";
          }
        if (option != 0)
         {
           cout<<"\n----------------------------------------"//
                "----------------\n";
         cout<<"\nDo you want to continue - 1 or EXIT the program - 0 : ";
         cin>>option;
         }


          }while(option != 0);

 cout<<"\n\tEND\n";

 return 0;

}
```

SAMPLE INPUT-OUTPUT



```
        STACK using ARRAY
__________________________________________________________

Enter the space to be provided in the stack : 10
Enter Number of elements in exsisting Stack : 9
Enter the elements : 23 45 11 23 54 8 9 0 1

---------------------------------------------------------

Operation on STACK :
  1. PUSH()
  2. POP()
  3. DISPLAY
  4. Exit
 ==> 1

Enter the number to PUSH : 10

---------------------------------------------------------
        Element PUSHED to the STACK
---------------------------------------------------------

Do you want to continue - 1 or EXIT the program - 0 : 1

---------------------------------------------------------

Operation on STACK :
  1. PUSH()
  2. POP()
  3. DISPLAY
  4. Exit
 ==> 3

The Elements in the Stack :
  23  45  11  23  54  8  9  0  1  10

---------------------------------------------------------

Do you want to continue - 1 or EXIT the program - 0 : 1

---------------------------------------------------------

Operation on STACK :
  1. PUSH()
  2. POP()
  3. DISPLAY
  4. Exit
 ==> 2

Pop off the element :
---------------------------------------------------------
10
---------------------------------------------------------

Do you want to continue - 1 or EXIT the program - 0 : 1

---------------------------------------------------------

Operation on STACK :
  1. PUSH()
  2. POP()
  3. DISPLAY
  4. Exit
 ==> 1

Enter the number to PUSH : 11

---------------------------------------------------------
        Element PUSHED to the STACK
---------------------------------------------------------

Do you want to continue - 1 or EXIT the program - 0 : 1

---------------------------------------------------------

Operation on STACK :
  1. PUSH()
  2. POP()
  3. DISPLAY
  4. Exit
 ==> 1
```



```
Enter the number to PUSH : 12

---------------------------------------------------------
        OVERFLOW
---------------------------------------------------------

Do you want to continue - 1 or EXIT the program - 0 : 1

---------------------------------------------------------

Operation on STACK :
  1. PUSH()
  2. POP()
  3. DISPLAY
  4. Exit
 ==> 3

The Elements in the Stack :
  23  45  11  23  54  8  9  0  1  11

---------------------------------------------------------

Do you want to continue - 1 or EXIT the program - 0 : 1

---------------------------------------------------------

Operation on STACK :
  1. PUSH()
  2. POP()
  3. DISPLAY
  4. Exit
 ==> 2

Pop off the element :
---------------------------------------------------------
11
---------------------------------------------------------

Do you want to continue - 1 or EXIT the program - 0 : 1

---------------------------------------------------------

Operation on STACK :
  1. PUSH()
  2. POP()
  3. DISPLAY
  4. Exit
 ==> 4

        END
```

OPERATION ON STACK USING LINKED LIST

AIM

To understand the PUSH() and POP() function using the STACK Data Structure constructed using the linked list.

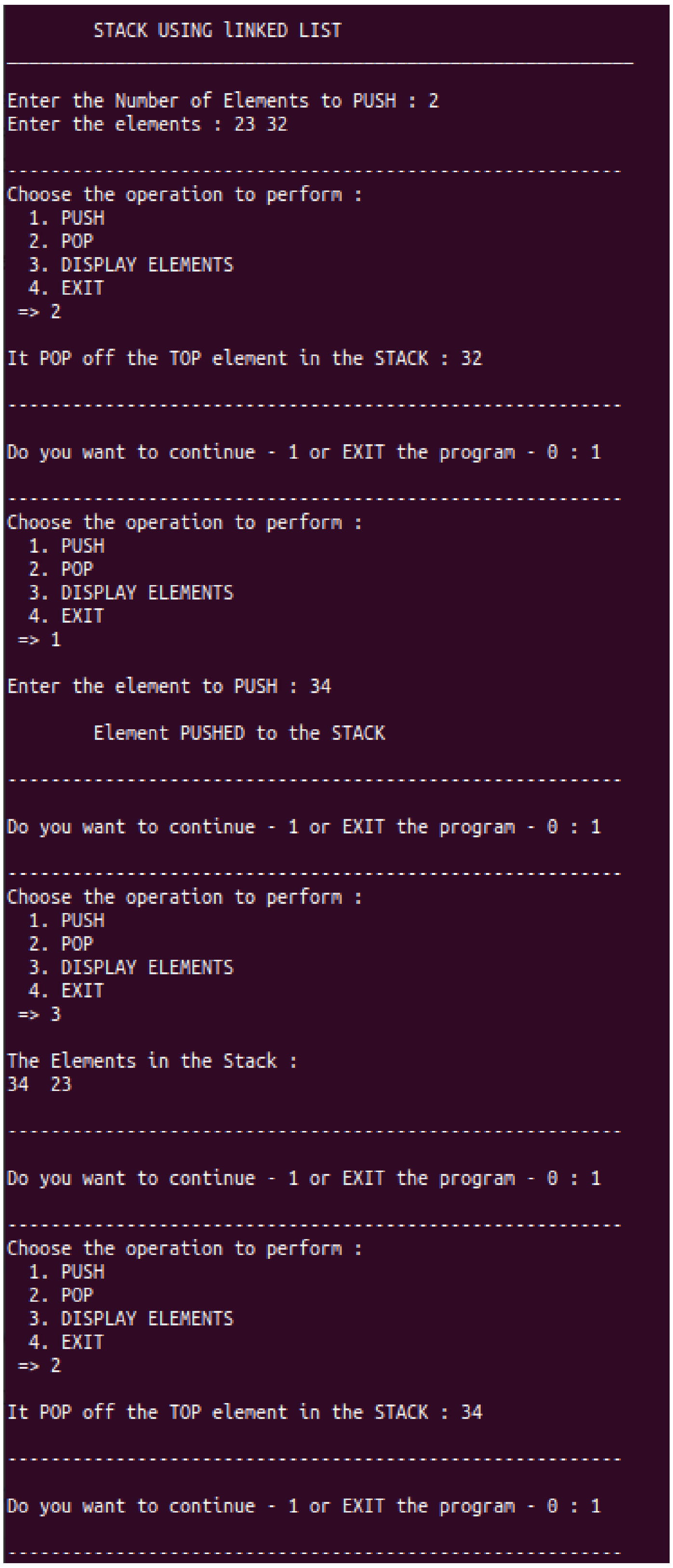
ALGORITHM

```
PUSH ( ITEM )
    1.  Create NEW node
    2.  if NEW == NULL then
            (i)     Print "OVERFLOW"
            (ii)    RETURN 1
    3.  End if condition
    4.  Set NEW->INFO = ITEM
    5.  Set NEW->LINK = TOP
    6.  Set TOP = NEW
    7.  RETURN 0
    8.  DELETE NEW
    9.  Exit

POP ( )
    1.  Create NEW node
    2.  if TOP == NULL then
            (i)     print "UNDERFLOW"
            (ii)    RETURN -1
    3.  End if condition
    4.  Set ITEM = TOP->INFO
    5.  Set TEMP = TOP
    6.  Set TOP = TOP->LINK
    7.  RETURN ITEM
    8.  DELETE TEMP
    9.  Exit
```

PROGRAM

```
#include<iostream>
using namespace std;

class Node
{
 public:
   int info;
   Node *link;
};
class STACKLINK
{
 Node *TOP;
 public:
   STACKLINK()
   {
   TOP = NULL;
   }
   int PUSH( int ITEM)
   {
    Node *NEW = new Node;
    if (NEW == NULL)
    {
     cout<<"\tOVERFLOW\n";
     return 1;
    }
    NEW->info = ITEM;
    NEW->link = TOP;
    TOP = NEW;
    return 0;
    delete NEW;
   }
   int POP()
   {
    Node *TEMP = new Node;
    if (TOP == NULL)
    {
     cout<<"UNDERFLOW\n";
     return -1;
    }
    int ITEM = TOP->info;
    TEMP = TOP;
    TOP = TOP->link;
     return ITEM;
     delete TEMP;
   }
   void DISPLAY()
   {
     Node *TEMP = new Node;
     TEMP = TOP;
     while (TEMP != NULL)
     {
      cout<<TEMP->info<<"  ";
      TEMP = TEMP->link;
     }
     cout<<endl;
     delete TEMP;
   }
};

int main()
{
 int MAXSTACK,terms,element,option,choice,key;
           cout<<"\n\tSTACK USING lINKED LIST\n";
           cout<<"__________________________________________________________\n";
          STACKLINK SL;

         cout<<"\nEnter the Number of Elements to PUSH : ";
           cin>>terms;
           cout<<"Enter the elements : ";
           for (int i=0 ;i<terms;i++)
           {
              cin>>element;
              SL.PUSH(element);
           }

           do
           { cout<<"\n-----------------------------------------"//
                        "-----------------\n";
             cout<<"Choose the operation to perform : \n  1. PUSH\n  "//
                        "2. POP\n  3. DISPLAY ELEMENTS \n  4. EXIT \n => ";
             cin>>option;

             switch(option)
             {
               case 1:
                {  cout<<"\nEnter the element to PUSH : ";
                   cin>>element;
                   key = SL.PUSH(element);
                   if(key != 1)
                   {
                      cout<<"\n\tElement PUSHED to the STACK\n";
                   }

                   break;
                }
              case 2:
                {   cout<<"\nIt POP off the TOP element in the STACK : ";
                    element = SL.POP();
       if (element != -1)
                     {
   cout<<element<<endl;
      }
      break;
 }
      case 3:
         {   cout<<"\nThe Elements in the Stack : \n";

              SL.DISPLAY();
              break;
         }
      case 4:
              break;
      default : cout<<"\nWrong Choice\n";
     }
  if (option !=4)
  {
          cout<<"\n-------------------------------------"//
                           "-------------------\n";
       cout<<"\nDo you want to continue - 1 or EXIT the program - 0 : ";
           cin>>option;
          }
     }while(option !=0 && option != 4);

 cout<<"\n\tEND\n";
 return 0;
}
```

SAMPLE INPUT-OUTPUT



```
        STACK USING lINKED LIST
___________________________________________________________

Enter the Number of Elements to PUSH : 2
Enter the elements : 23 32

---------------------------------------------------------
Choose the operation to perform :
  1. PUSH
  2. POP
  3. DISPLAY ELEMENTS
  4. EXIT
 => 2

It POP off the TOP element in the STACK : 32

---------------------------------------------------------

Do you want to continue - 1 or EXIT the program - 0 : 1

---------------------------------------------------------
Choose the operation to perform :
  1. PUSH
  2. POP
  3. DISPLAY ELEMENTS
  4. EXIT
 => 1

Enter the element to PUSH : 34

        Element PUSHED to the STACK

---------------------------------------------------------

Do you want to continue - 1 or EXIT the program - 0 : 1

---------------------------------------------------------
Choose the operation to perform :
  1. PUSH
  2. POP
  3. DISPLAY ELEMENTS
  4. EXIT
 => 3

The Elements in the Stack :
34  23

---------------------------------------------------------

Do you want to continue - 1 or EXIT the program - 0 : 1

---------------------------------------------------------
Choose the operation to perform :
  1. PUSH
  2. POP
  3. DISPLAY ELEMENTS
  4. EXIT
 => 2

It POP off the TOP element in the STACK : 34

---------------------------------------------------------

Do you want to continue - 1 or EXIT the program - 0 : 1

---------------------------------------------------------
```

BINARY SEARCH TREE

AIM

To understand the construction and working of Binary search Tree associated with the number. Here the SEARCH() function helps to search the number as per the tree construction, INSERT() function help to insert the number input by the user in the correct position as per the Binary Search Tree conditions, DELETE() helps to delete a particular node and replace it with the proper number from the sub trees, and the PREORDER(), POSTORDER() and INORDER() traversal functions helps in traversal through the Binary tree.

ALGORITHM

```
PREORDER ( NODE )
    1.  if NODE == NULL then
            (i) RETURN
    2.  end if
    3.  print NODE->INFO
    4.  PERORDER (NODE->LEFT)
    5.  PERORDER (NODE->RIGHT)
    6.  Exit

INORDER ( NODE )
    1.  if NODE == NULL then
            (i) RETURN
    2.  end if
    3.  INORDER (TEMP->LEFT)
    4.  print NODE->INFO
    5.  INORDER (TEMP->RIGHT)
    6.  Exit

POSTORDER ( NODE )
    1.  if  NODE == NULL then
            (i) RETURN
    2.  end if
    3.  POSTORDER (TEMP->LEFT)
    4.  POSTORDER (TEMP->RIGTH)
    5.  print NODE->INFO
    6. Exit

INSERT ( N_NODE , NODE)
```

```
    1.  Create NEW_NODE
    2.  if ROOT == NODE then
            (i)     Set ROOT = NEW_NODE
            (ii)    RETURN ROOT
    3.  Exit
    4.  if NEW->INFO < ROOT->INFO then
            (i)     Set ROOT->LEFT = INSERT(ROOT->LEFT , NEW_NODE)
    5.  else if NEW->INFO > ROOT->INFO then
            (i)     Set ROOT->RIGHT = INSERT(ROOT->RIGHT , NEW_NODE)
    6.  else
            (i)     RETURN ROOT
    7.  end if
    8.  RETURN ROOT
    9.  Exit

SEARCH ( DATA )
    1.  if ROOT == NULL then
    2.      (i)     RETURN ROOT
    3.  else
            (i)     Create TEMP NODE and
            (ii)    Repeat the steps while (TEMP != NULL)
                        (a) if DATA == TEMP->INFO
                                *   print DATA located
                                *   RETURN TEMP
                        (b) else if DATA < TEMP->INFO
                                *  Set TEMP = TEMP->LEFT
                        (c) else
                                *  Set TEMP = TEMP->RIGHT
                        (d) End
            (iii)   End while loop
    4.  End if condition
    5.  RETURN NULL
    6.  Exit

DELETE ( N_TEMP , DATA )
    1.  Create TEMP_NODE
    2.  if TEMP_NODE == NULL
            (i)     RETURN NULL
    3.  else if DATA < TEMP_NODE->INFO
            (i)    Set TEMP_NODE->LEFT = DELETE(TEMP_NODE->LEFT, DATA)
    4.  else if DATA > TEMP_NODE->INFO
```

```
                (i)     Set TEMP_NODE->RIGHT = DELETE(TEMP_NODE->RIGHT , DATA)
        5.  else
                (i)     if TEMP_NODE->LEFT == NULL then
                            (a) Create NEW_NODE
                            (b) Set NEW_NODE = TEMP_NODE->RIGHT
                            (c) DELETE TEMP_NODE
                            (d) RETURN NEW_NODE
                (ii)    else if TEMP_NODE->RIGHT == NULL then
                            (a) Create NEW_NODE
                            (b) Set NEW_NODE = TEMP_NODE->LEFT
                            (c) DELETE TEMP_NODE
                            (d) RETURN NEW_NODE
                (iii)   else
                            (a) Create NEW_NODE= = MINVALUE(TEMP_NODE->RIGHT)
                            (b) Set TEMP_NODE->INFO = NEW_NODE->INFO
                            (c) Set TEMP_NODE->RIGHT = DELETE( TEMP_NODE->RIGHT
                            , NEW_NODE->INFO)
                            (d) TEMP_NODE->RIGHT = DELETE(TEMP->RIGHT
                            , NEW_NODE->INFO)
                (iv)    End if condition
        6.  End if condition
        7.RETURN TEMP
        8. Exit

    MINVALUE( NODE )
        1.  Create NEW_NODE
        3.  Set NEW_NODE = NODE
        2.  Repeat the step while NEW->LEFT = NULL
                (i)   NEW = NEW->LEFT
        4.  End while loop
        5.  RETURN NEW
        6.  Exit
```

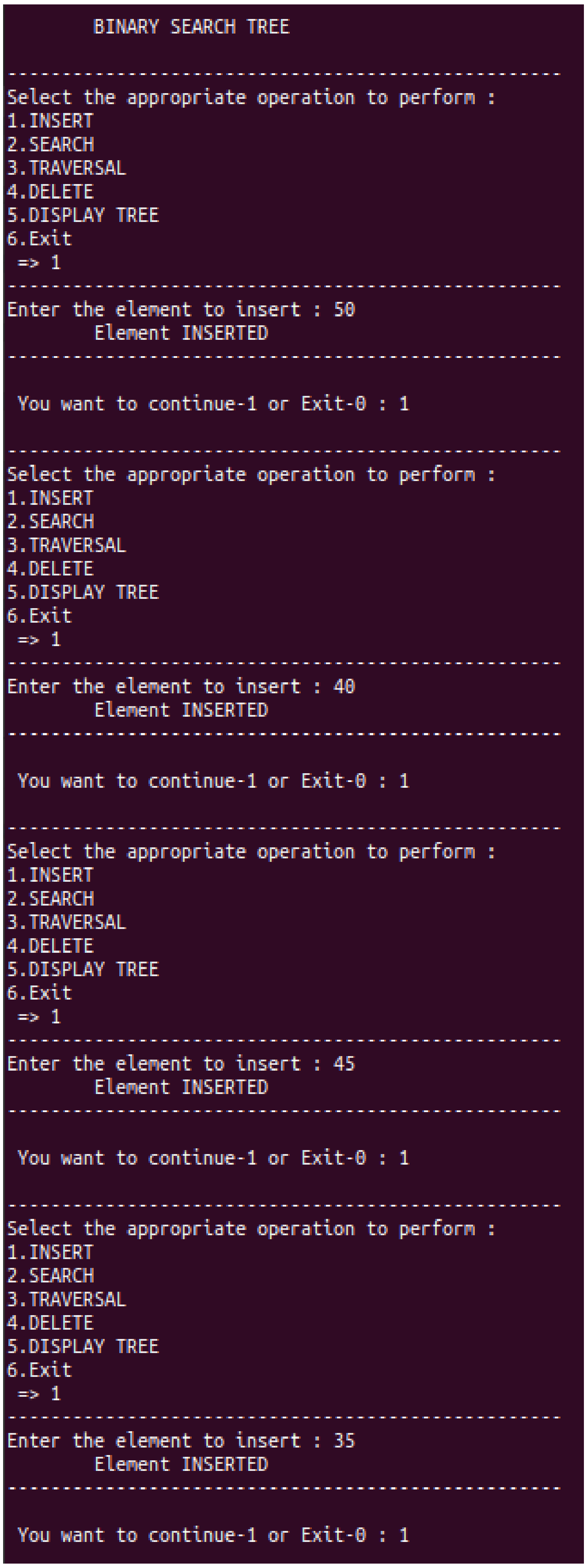
PROGRAM

```
#include <iostream>
using namespace std;

class Node
{
 public:
  int info;
  Node* left;
  Node* right;
  int successor = 1;
  Node()
  {
   info = 0;
   left = NULL;
   right = NULL;
  }
  Node(int Value )
  {
   info = Value;
   left = 0;
   right = 0;
  }
};

class BST
{
 public:
  Node* ROOT;
  BST()
  {
   ROOT = 0;
  }

  Node* INSERT(Node* root, Node* NEW)
  {
   if( root==NULL)
   {
     root = NEW;
     cout<<"\tElement INSERTED";
     return root;
   }

   if( NEW->info < root->info)
   {
     root->left = INSERT(root->left , NEW);
   }
   else if ( NEW->info > root->info)
   {
     root->right = INSERT( root->right , NEW);
   }
   else
   {
    cout<<"\nValue already exsist\n";
    return root;
   }

   return root;
  }

  void PREORDER(Node* TEMP)
  {
    if( TEMP == NULL)
          return;

    cout<<TEMP->info<<"  ";
    PREORDER(TEMP->left);
    PREORDER(TEMP->right);
  }

  void INORDER(Node* TEMP)
  {
    if( TEMP == NULL)
          return;

    INORDER(TEMP->left);
    cout<<TEMP->info<<"  ";
    INORDER(TEMP->right);
  }

  void POSTORDER(Node* TEMP)
  {
    if( TEMP == NULL)
          return;

    POSTORDER(TEMP->left);
    POSTORDER(TEMP->right);
    cout<<TEMP->info<<"  ";
  }
```

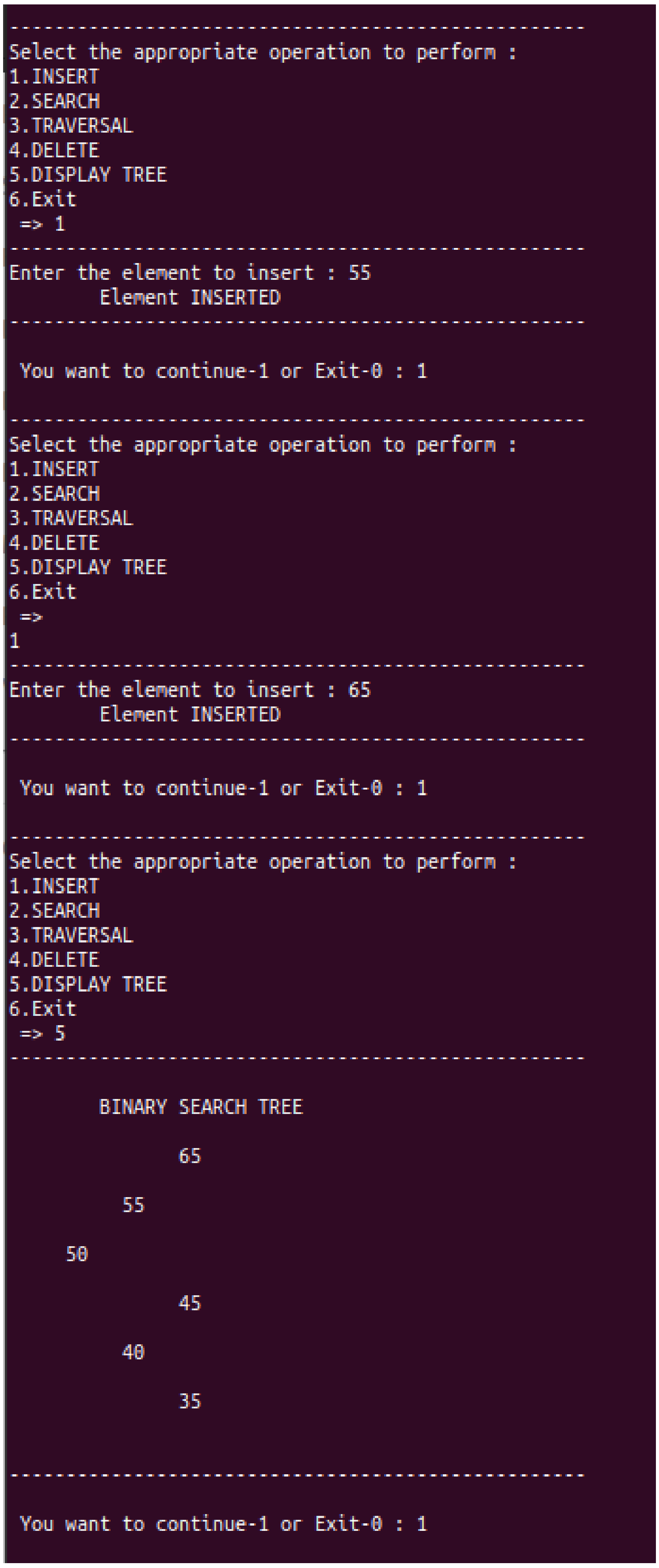
```
Node* SEARCH(int DATA)
{
  if(ROOT == NULL)
    {
      cout<<"No Elements in the Tree\n";
      return ROOT;
    }
  else
    {
      Node* TEMP = ROOT;
      while( TEMP != NULL)
      {
       if( DATA == TEMP->info)
       {
         cout<<TEMP->info<<" located  \n";
         return TEMP;
       }
       else if(DATA < TEMP->info)
       {
         TEMP = TEMP->left;
       }
       else
       {
        TEMP = TEMP->right;
       }
      }
    }
    cout<<"\nElement not found\n";
    return NULL;
}

Node* MINValue(Node* NEW)
{
 Node* THIS = NEW;
 while (THIS->left != NULL)
 {
  THIS = THIS->left;
 }
 return THIS;
}

Node* DELETE(Node* TEMP,int DATA)
{
  if(TEMP == NULL)
  {
   return NULL;
  }
  else if( DATA < TEMP->info)
  {
   TEMP->left = DELETE(TEMP->left , DATA);
  }
  else if(DATA > TEMP->info)
  {
   TEMP->right = DELETE(TEMP->right , DATA);
  }
  else
  {
   if( TEMP->left == NULL)
   {
     Node* NEW1 = TEMP->right;
     delete TEMP;
     return NEW1;
   }
   else if(TEMP->right == NULL)
   {
     Node* NEW1 = TEMP->left;
     delete TEMP;
     return NEW1;
   }
   else
   {
     Node* NEW1 = MINValue(TEMP->right);
     TEMP->info = NEW1->info;
     TEMP->right = DELETE(TEMP->right , NEW1->info);
   }
  }
  return TEMP;
}

void PRINT(Node* TEMP, int SPACE)
```

```
  {
    if (TEMP == NULL)
          return;
    SPACE = SPACE+5;
    PRINT( TEMP->right, SPACE);
    cout<<"\n";
    for (int i=20;i<SPACE;i++)
    {
      cout<<" ";
    }
    cout<<TEMP->info<<"\n";
    PRINT(TEMP->left,SPACE);
  }
};

int main()
{
  cout<<"\n\tBINARY SEARCH TREE\n";
  int choice , data , option=0 ;
  BST TREE;

  do
  {
     cout<<"\n--------------------------------------------------";
     cout<<"\nSelect the appropriate operation to perform : \n"/*
     */"1.INSERT \n2.SEARCH\n3.TRAVERSAL\n4.DELETE\n5.DISPLAY TREE"//
                "\n6.Exit\n => ";
     cin>>choice;
     Node* TEMP= new Node;
     if(TEMP == NULL)
     {
      cout<<"\n********UNDERFLOW*********\n";
      choice=5;
     }
     else
     {
     cout<<"-------------------------------------------------\n";
  switch(choice)
  {
   case 1:
     cout<<"Enter the element to insert : ";
     cin>>data;
     TEMP->info = data;
     TREE.ROOT = TREE.INSERT(TREE.ROOT,TEMP);
     break;

   case 2:
     cout<<"\nEnter the element to search : ";
     cin>>data;
     TEMP = TREE.SEARCH(data);
     break;

   case 3:
     cout<<"\nChoose the TRAVERSAL technique\n\t1.PRE-ORDER TRAVERSAL\n"//
            "\t2.INORDER TRAVERSAL\n\t3.POST-ORDER TRAVERSAL\n\t ::> ";
     cin>>choice;

     switch(choice)
     {
       case 1:
           cout<<"\n\tPRE-ORDER TRAVERSAL : ";
           TREE.PREORDER(TREE.ROOT);
           cout<<endl;
           break;

       case 2:
           cout<<"\n\tINORDER TRAVERSAL : ";
           TREE.INORDER(TREE.ROOT);
           cout<<endl;
           break;

       case 3:
           cout<<"\n\tPOST-ORDER TRAVERSAL : ";
           TREE.POSTORDER(TREE.ROOT);
           cout<<endl;
           break;

       default:cout<<"\tWrong choice\n";
     }
     break;

   case 4:
     cout<<"\nEnter the number to delete : ";
     cin>>data;
     TEMP = TREE.SEARCH(data);
     if (TEMP != NULL)
     {
            TREE.DELETE(TREE.ROOT, data);
            cout<<"\nValue deleted.\n";
     }
     else
     {
            cout<<"\nValue not found\n";
     }
     break;

   case 5:
       cout<<"\n\tBINARY SEARCH TREE\n";
       TREE.PRINT(TREE.ROOT, 20);
       cout<<endl;
       break;

   case 6:
       break;

   default:cout<<"\n\tWrong choice\n";
  }
  }
  if(choice != 6)
  {
    cout<<"\n--------------------------------------------------\n";
    cout<<"\n You want to continue-1 or Exit-0 : ";
    cin>>option;
    if (option == 0)
    {
       choice = 6;
    }
  }
  }while(choice != 6 );

  cout<<"\n\tEND\n";
  return 0;
}
```

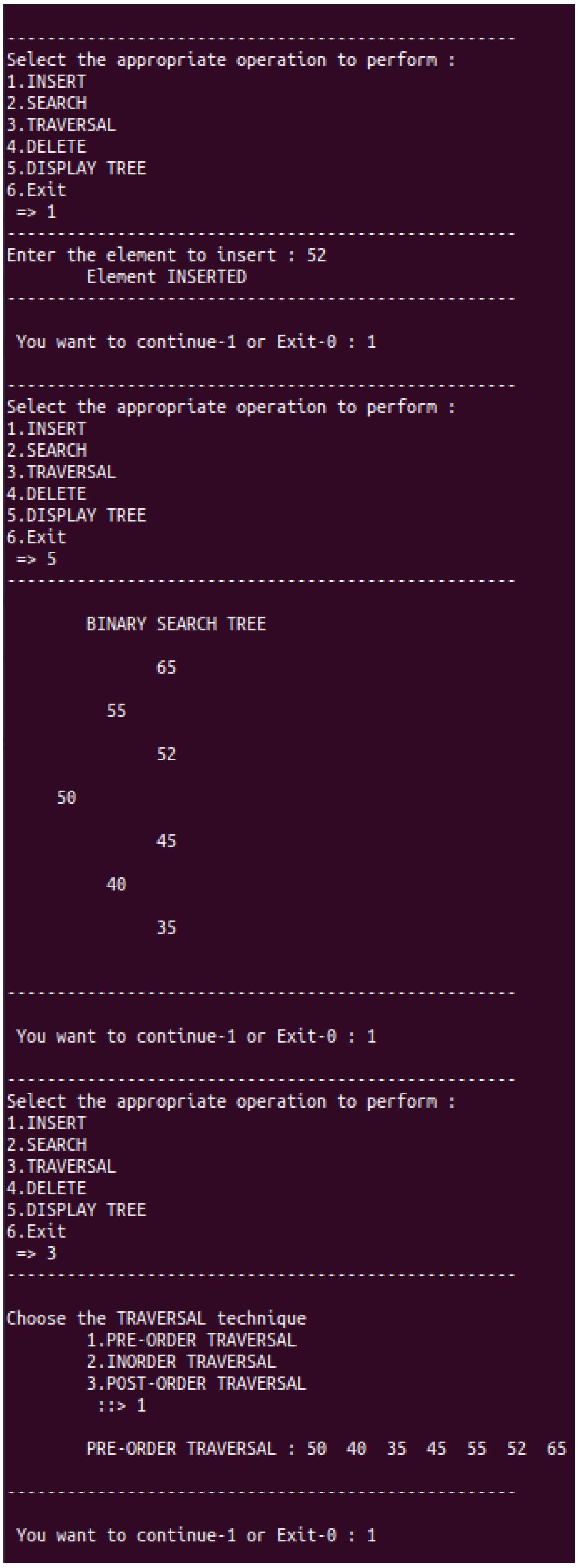
SAMPLE INPUT-OUTPUT



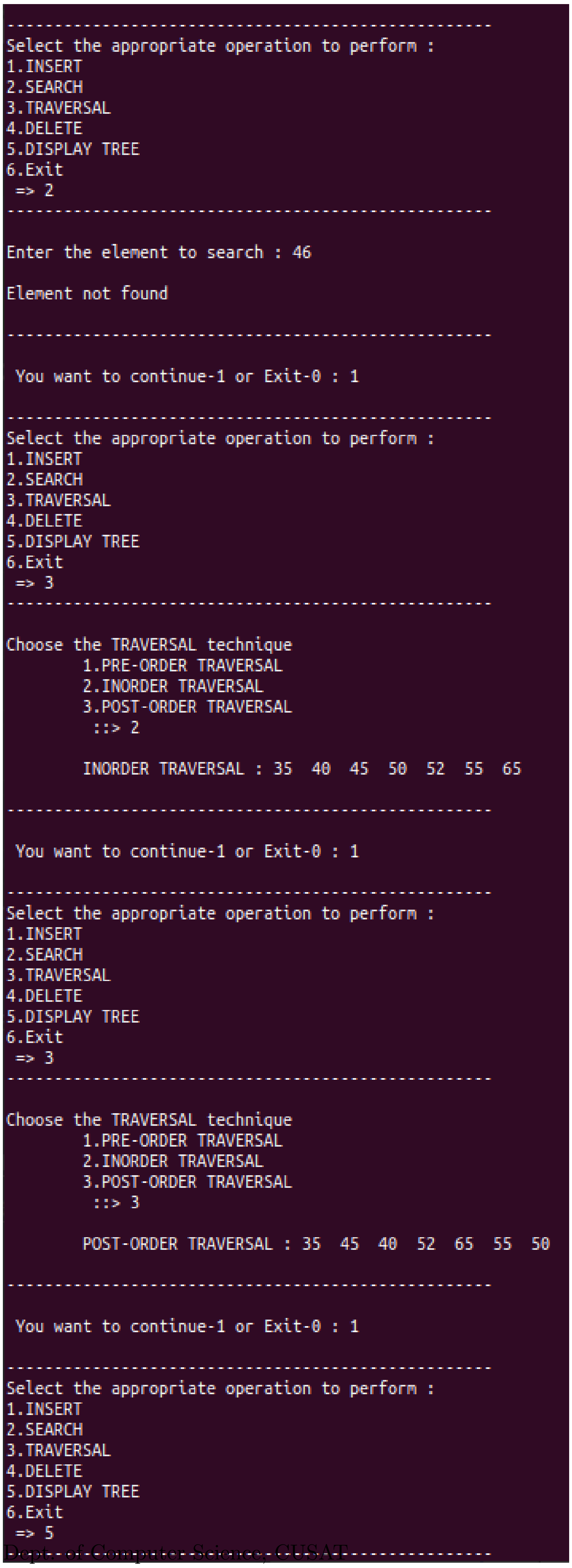
```
        BINARY SEARCH TREE

---------------------------------------------------
Select the appropriate operation to perform :
1.INSERT
2.SEARCH
3.TRAVERSAL
4.DELETE
5.DISPLAY TREE
6.Exit
 => 1
---------------------------------------------------
Enter the element to insert : 50
        Element INSERTED
---------------------------------------------------

 You want to continue-1 or Exit-0 : 1

---------------------------------------------------
Select the appropriate operation to perform :
1.INSERT
2.SEARCH
3.TRAVERSAL
4.DELETE
5.DISPLAY TREE
6.Exit
 => 1
---------------------------------------------------
Enter the element to insert : 40
        Element INSERTED
---------------------------------------------------

 You want to continue-1 or Exit-0 : 1

---------------------------------------------------
Select the appropriate operation to perform :
1.INSERT
2.SEARCH
3.TRAVERSAL
4.DELETE
5.DISPLAY TREE
6.Exit
 => 1
---------------------------------------------------
Enter the element to insert : 45
        Element INSERTED
---------------------------------------------------

 You want to continue-1 or Exit-0 : 1

---------------------------------------------------
Select the appropriate operation to perform :
1.INSERT
2.SEARCH
3.TRAVERSAL
4.DELETE
5.DISPLAY TREE
6.Exit
 => 1
---------------------------------------------------
Enter the element to insert : 35
        Element INSERTED
---------------------------------------------------

 You want to continue-1 or Exit-0 : 1
```



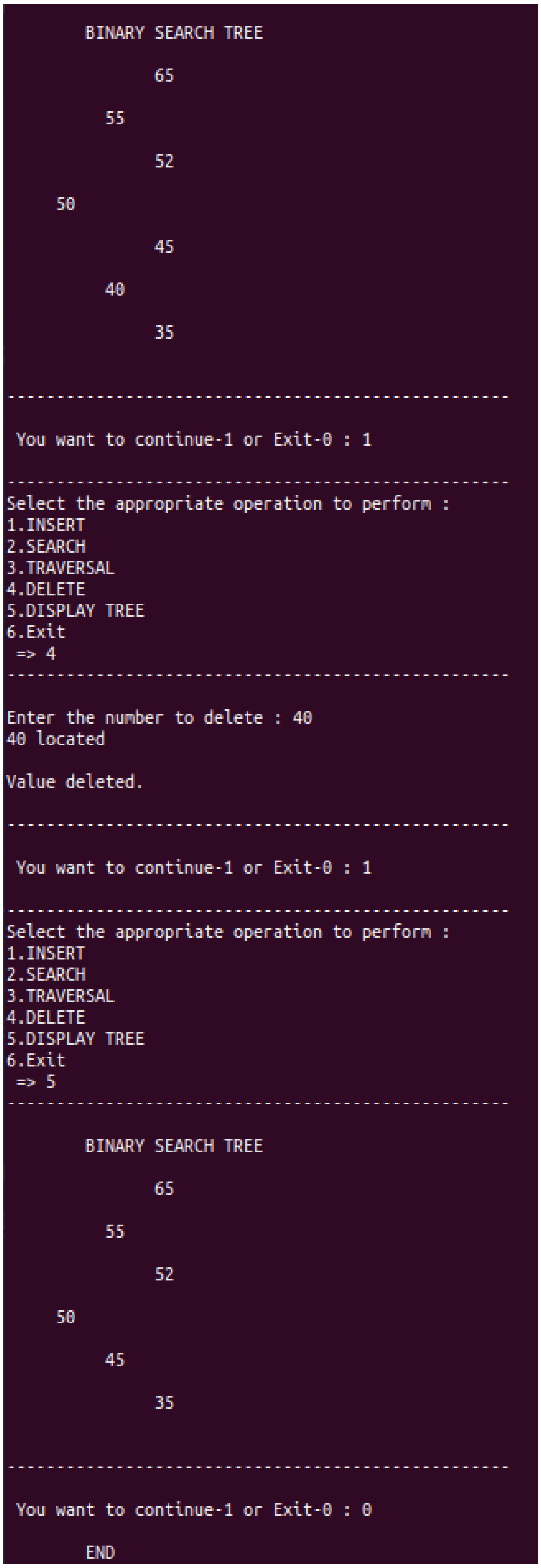
```
---------------------------------------------------
Select the appropriate operation to perform :
1.INSERT
2.SEARCH
3.TRAVERSAL
4.DELETE
5.DISPLAY TREE
6.Exit
 => 1
---------------------------------------------------
Enter the element to insert : 55
        Element INSERTED
---------------------------------------------------

 You want to continue-1 or Exit-0 : 1

---------------------------------------------------
Select the appropriate operation to perform :
1.INSERT
2.SEARCH
3.TRAVERSAL
4.DELETE
5.DISPLAY TREE
6.Exit
 =>
1
---------------------------------------------------
Enter the element to insert : 65
        Element INSERTED
---------------------------------------------------

 You want to continue-1 or Exit-0 : 1

---------------------------------------------------
Select the appropriate operation to perform :
1.INSERT
2.SEARCH
3.TRAVERSAL
4.DELETE
5.DISPLAY TREE
6.Exit
 => 5
---------------------------------------------------

        BINARY SEARCH TREE

               65

          55

     50

               45

          40

               35


---------------------------------------------------

 You want to continue-1 or Exit-0 : 1
```



```
---------------------------------------------------
Select the appropriate operation to perform :
1.INSERT
2.SEARCH
3.TRAVERSAL
4.DELETE
5.DISPLAY TREE
6.Exit
 => 1
---------------------------------------------------
Enter the element to insert : 52
        Element INSERTED
---------------------------------------------------

 You want to continue-1 or Exit-0 : 1

---------------------------------------------------
Select the appropriate operation to perform :
1.INSERT
2.SEARCH
3.TRAVERSAL
4.DELETE
5.DISPLAY TREE
6.Exit
 => 5
---------------------------------------------------

        BINARY SEARCH TREE

               65

          55

               52

     50

               45

          40

               35


---------------------------------------------------

 You want to continue-1 or Exit-0 : 1

---------------------------------------------------
Select the appropriate operation to perform :
1.INSERT
2.SEARCH
3.TRAVERSAL
4.DELETE
5.DISPLAY TREE
6.Exit
 => 3
---------------------------------------------------

Choose the TRAVERSAL technique
        1.PRE-ORDER TRAVERSAL
        2.INORDER TRAVERSAL
        3.POST-ORDER TRAVERSAL
         ::> 1

        PRE-ORDER TRAVERSAL : 50  40  35  45  55  52  65

---------------------------------------------------

 You want to continue-1 or Exit-0 : 1
```



```
--------------------------------------------------
Select the appropriate operation to perform :
1.INSERT
2.SEARCH
3.TRAVERSAL
4.DELETE
5.DISPLAY TREE
6.Exit
 => 2
--------------------------------------------------

Enter the element to search : 46

Element not found

--------------------------------------------------

 You want to continue-1 or Exit-0 : 1

--------------------------------------------------
Select the appropriate operation to perform :
1.INSERT
2.SEARCH
3.TRAVERSAL
4.DELETE
5.DISPLAY TREE
6.Exit
 => 3
--------------------------------------------------

Choose the TRAVERSAL technique
        1.PRE-ORDER TRAVERSAL
        2.INORDER TRAVERSAL
        3.POST-ORDER TRAVERSAL
         ::> 2

        INORDER TRAVERSAL : 35  40  45  50  52  55  65

--------------------------------------------------

 You want to continue-1 or Exit-0 : 1

--------------------------------------------------
Select the appropriate operation to perform :
1.INSERT
2.SEARCH
3.TRAVERSAL
4.DELETE
5.DISPLAY TREE
6.Exit
 => 3
--------------------------------------------------

Choose the TRAVERSAL technique
        1.PRE-ORDER TRAVERSAL
        2.INORDER TRAVERSAL
        3.POST-ORDER TRAVERSAL
         ::> 3

        POST-ORDER TRAVERSAL : 35  45  40  52  65  55  50

--------------------------------------------------

 You want to continue-1 or Exit-0 : 1

--------------------------------------------------
Select the appropriate operation to perform :
1.INSERT
2.SEARCH
3.TRAVERSAL
4.DELETE
5.DISPLAY TREE
6.Exit
 => 5
--------------------------------------------------
```



```
        BINARY SEARCH TREE

               65

          55

               52

     50

               45

          40

               35


---------------------------------------------------

 You want to continue-1 or Exit-0 : 1

---------------------------------------------------
Select the appropriate operation to perform :
1.INSERT
2.SEARCH
3.TRAVERSAL
4.DELETE
5.DISPLAY TREE
6.Exit
 => 4
---------------------------------------------------

Enter the number to delete : 40
40 located

Value deleted.

---------------------------------------------------

 You want to continue-1 or Exit-0 : 1

---------------------------------------------------
Select the appropriate operation to perform :
1.INSERT
2.SEARCH
3.TRAVERSAL
4.DELETE
5.DISPLAY TREE
6.Exit
 => 5
---------------------------------------------------

        BINARY SEARCH TREE

               65

          55

               52

     50

          45

               35


---------------------------------------------------

 You want to continue-1 or Exit-0 : 0

        END
```

QUICK SORT

AIM

To understand the sorting of the numbers using the Quick Sort algorithm

ALGORITHM

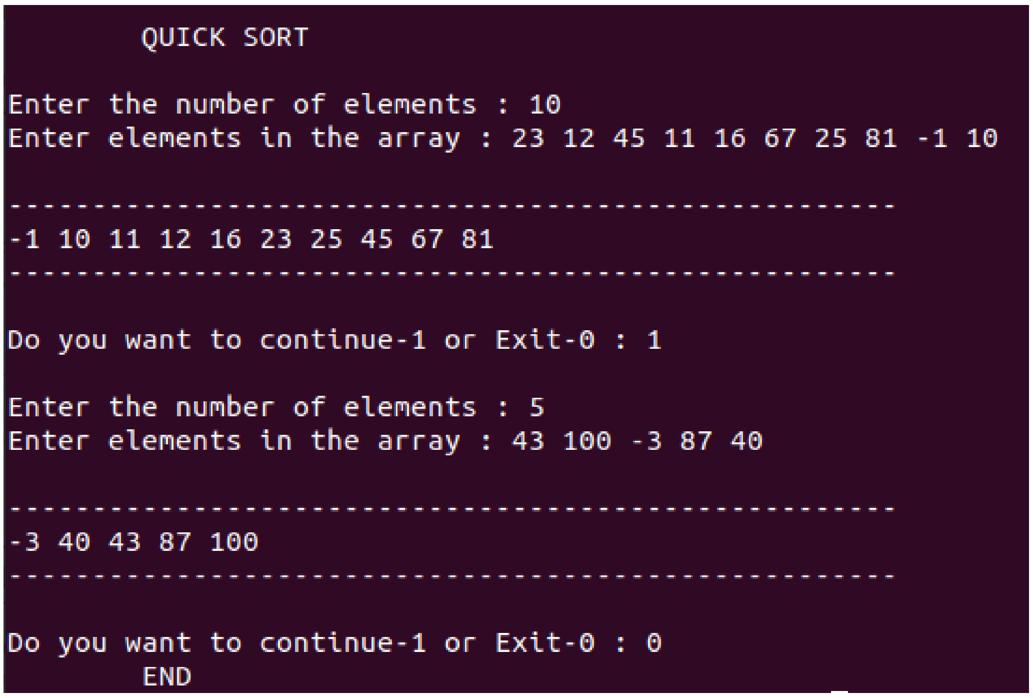
```
 QUICK(A ,N ,BEG ,END ,LOC)
    1. set LEFT = BEG , RIGHT = END , LOC =  BEG
    2. Repeat while A[LOC] < = A[RIGHT] and LOC ! = RIGHT
        a. RIGHT = RIGHT { 1
        b. if LOC = RIGHT then return
        c. if A[LOC] > A[RIGHT] then
    i. TEMP = A[LOC]
A[LOC] = A[RIGHT]
A[RIGHT] = TEMP

    ii. LOC = RIGHT

    iii. Go to step 3.



    3. Repeat while A[LOC] < = A[LEFT] and LOC !  =LEFT
    a. LEFT = LEFT + 1
    b. if LOC = LEFT then return
    c. if A[LOC] > A[LEFT] then
    i. TEMP = A[LOC]
A[LOC] = A[LEFT]
A[LEFT] = TEMP

    ii. LOC = LEFT
    iii. Go to step 2.

    4. Exit
```

PROGRAM

```
#include<iostream>
using namespace std;

void swapping(int &a, int &b)
{
   int temp;
   temp = a;
   a = b;
   b = temp;
}

void display(int *array, int size)
{
   for(int i = 0; i<size; i++)
      cout << array[i] << " ";
   cout << endl;
}

int partition(int *array, int lower, int upper)
{

   int pivot, start, end;
   pivot = array[lower];
   start = lower; end = upper;

   while(start < end) {
      while(array[start] <= pivot && start<end)
      {
         start++;
      }

      while(array[end] > pivot)
      {
         end--;
      }

      if(start < end) {
         swap(array[start], array[end]);
      }
   }

   array[lower] = array[end];
   array[end] = pivot;
   return end;
}
```

```
void quickSort(int *array, int left, int right)
{
   int q;

   if(left < right)
   {
      q = partition(array, left, right);
      quickSort(array, left, q-1);
      quickSort(array, q+1, right);

   }
}

int main()
{
   int n, option;
   cout<<"\n\tQUICK SORT\n";
   do
   {
   cout << "\nEnter the number of elements : ";
   cin >> n;
   int arr[n];
   cout << "Enter elements in the array : " ;

   for(int i = 0; i<n; i++)
   {
      cin >> arr[i];
   }

   cout<<"\n-----------------------------------------------------\n";
   quickSort(arr, 0, n-1);
   display(arr, n);
      cout<<"-----------------------------------------------------\n";
     cout<<"\nDo you want to continue-1 or Exit-0 : ";
     cin>>option;
     }while(option != 0);
     cout<<"\tEND\n";
      return 0;
}
```

SAMPLE INPUT-OUTPUT



```
        QUICK SORT

Enter the number of elements : 10
Enter elements in the array : 23 12 45 11 16 67 25 81 -1 10

-----------------------------------------------------
-1 10 11 12 16 23 25 45 67 81
-----------------------------------------------------

Do you want to continue-1 or Exit-0 : 1

Enter the number of elements : 5
Enter elements in the array : 43 100 -3 87 40

-----------------------------------------------------
-3 40 43 87 100
-----------------------------------------------------

Do you want to continue-1 or Exit-0 : 0
        END
```

OPERATION ON QUEUE USING ARRAY

AIM

To understand the insert or en-queue and deletion or de-queue in a queue. Demonstration using the function ENQUEUE() and DEQUEUE() functions. Its working using the data structure Array.

ALGORITHM

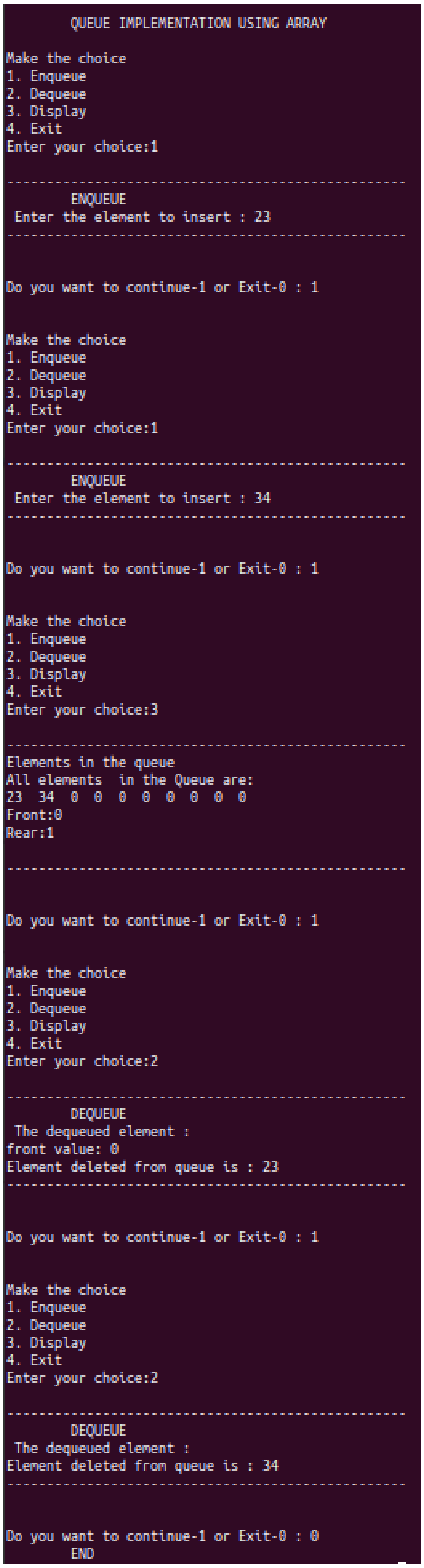
```
ENQUEUE ( ITEM )
    1.  if REAR = ARRAY_SIZE-1 then
            (i)     print "OVERFLOW"
            (ii)    return
    2.  else if FRONT == -1 && REAR == -1   then
            (i)     set REAR = 0
            (ii)    set FRONT = 0
            (iii)   set AARAY[REAR] = ITEM
    3.  else
            (i)     set REAR = REAR+1
            (ii)    set ARRAY[REAR] = ITEM
    4.  end if condition
    5.  Exit

DEQUEUE ( ITEM )
    1.  if  FRONT == -1 && REAR ==-1 then
            (i)     print "UNDERFLOW"
            (ii)    RETURN
    2.  else if REAR == FRONT then
            (i)     set ITEM = ARRAY[REAR]
            (ii)    set REAR = -1
            (iii)   set FRONT = -1
            (iv)    print "DELETED"
            (v)     FRONT = FRONT+1
    3.  else
            (i)     set ITEM = AARAY[FRONT]
            (ii)    set ARRAY[FRONT] = 0
            (iii)   print "DELETED
            (iv)    FRONT = FRONT+1
    4. end if condition
    5. Exit
```

PROGRAM

```
#include<iostream>
using namespace std;
int n=10;
class Queue
{
  private:
  int front;
  int rear;
  int arr[20];

  public:
    Queue()
    {
      front = -1;
      rear = -1;
      for (int i = 0; i < n; i++)
      {
        arr[i] = 0;
      }
    }
  void enqueue(int val)
  {
     if (rear == n-1)
   {
      cout << "Queue Overflow!!" << endl;
      return;
    }
    else if (front == -1 && rear == -1)
    {
      rear = 0;
      front = 0;
      arr[rear] = val;
    }
    else
    {
      rear++;
      arr[rear] = val;
    }
```

```
}

void dequeue()
{
  int val;
  if (front == -1 && rear == -1)
  {
     cout << "Queue Underflow!!" << endl;

  }
  else if (rear == front)
  {
    val = arr[rear];
    rear = -1;
    front = -1;
    cout<<"Element deleted from queue is : "<< val <<endl;
    front++;
  }
  else
  {
    cout << "front value: " << front << endl;

    val = arr[front];
    arr[front] = 0;

    cout<<"Element deleted from queue is : "<< val <<endl;

    front++;
  }
}

void display()
{
if (front==-1)
{
cout<<"Queue is empty!!"<<endl;
}
else
{
   cout << "All elements  in the Queue are:"<<endl;
   }
    for (int i = 0; i < n; i++)
    {
      cout << arr[i] << "  ";
    }
      cout<<"\nFront:"<<front<<endl;
      cout<<"Rear:"<<rear<<endl;
      cout<<endl;
  }
};

int main()
{
  Queue q1;

  int value, option,ch=1;

  cout<<"\n\tQUEUE IMPLEMENTATION USING ARRAY";

  do
  {

    cout << "\n\nMake the choice " << endl;
    cout << "1. Enqueue\n2. Dequeue \n3. Display\n4. Exit\n";
    cout<<"Enter your choice:";
    cin >> option;

    switch (option)
    {
    case 1:
    {
      cout<<"\n-----------------------------------------------\n";
      cout << "\tENQUEUE\n Enter the element to insert : ";
      cin >> value;

      q1.enqueue(value);

      break;
    }
    case 2:
    {
      cout<<"\n--------------------------------------------------\n";
      cout << "\tDEQUEUE\n The dequeued element : " <<  endl;

      q1.dequeue() ;

      break;
    }
    case 3:
    {
      cout<<"\n--------------------------------------------------\n";
      cout << "Elements in the queue" << endl;
      q1.display();
      break;
    }
    case 4:
    {
      ch = 0 ;
      break;
    }
    default:
    {
      cout<<"\n--------------------------------------------------\n";
      cout << "Wrong choice" << endl;
    }
    }

    if(ch!=0)
    {
    cout<<"--------------------------------------------------\n\n";
    cout<<"\nDo you want to continue-1 or Exit-0 : ";
    cin>>ch;
    }
  } while (ch != 0);

  cout<<"\tEND\n";

  return 0;
}
```

SAMPLE INPUT-OUTPUT



```
        QUEUE IMPLEMENTATION USING ARRAY

Make the choice
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice:1

--------------------------------------------------
        ENQUEUE
 Enter the element to insert : 23
--------------------------------------------------


Do you want to continue-1 or Exit-0 : 1


Make the choice
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice:1

--------------------------------------------------
        ENQUEUE
 Enter the element to insert : 34
--------------------------------------------------


Do you want to continue-1 or Exit-0 : 1


Make the choice
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice:3

--------------------------------------------------
Elements in the queue
All elements  in the Queue are:
23  34  0  0  0  0  0  0  0  0
Front:0
Rear:1

--------------------------------------------------


Do you want to continue-1 or Exit-0 : 1


Make the choice
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice:2

--------------------------------------------------
        DEQUEUE
 The dequeued element :
front value: 0
Element deleted from queue is : 23
--------------------------------------------------


Do you want to continue-1 or Exit-0 : 1


Make the choice
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice:2

--------------------------------------------------
        DEQUEUE
 The dequeued element :
Element deleted from queue is : 34
--------------------------------------------------


Do you want to continue-1 or Exit-0 : 0
        END
```

OPERATION ON QUEUE USING LINKED LIST

AIM

To understand the insert or en-queue and deletion or de-queue in a queue. Demonstration using the function ENQUEUE() and DEQUEUE() functions. Its working using the data structure Linked list.

ALGORITHM

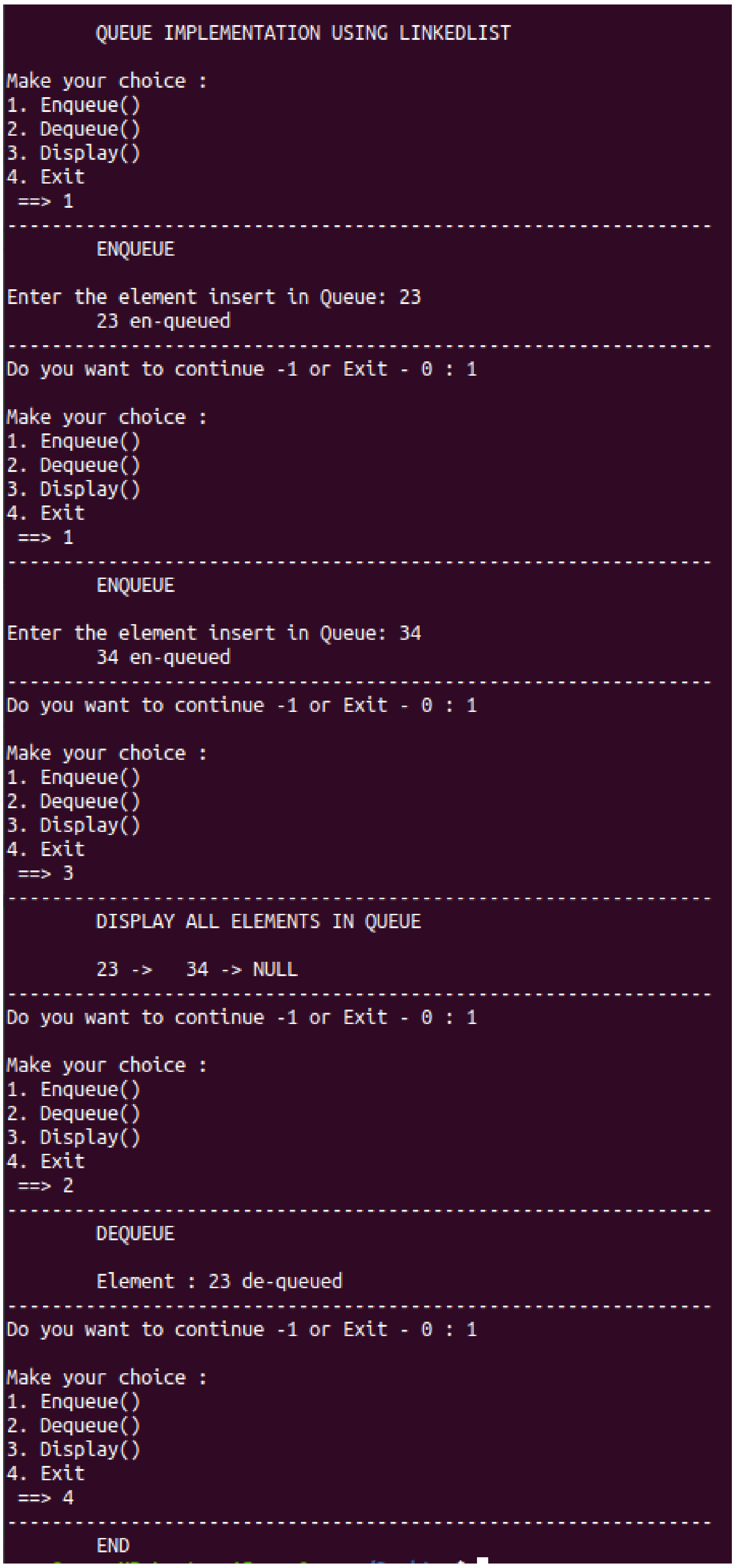
```
ENQUEUE ( ITEM )
    1.   Create NEW node odject of class NODE
    2.   if NEW == NULL then
            (i)     print "OVERFLOW"
            (ii)    return
    3.  end if condition
    4.  set NEW->INFO = ITEM
    5.  set NEW->LINK = NULL
    6.  if FRONT = NULL
            (i)     set FRONT = REAR
            (ii)    set REAR = TEMP
    7.  else
            (i)     set REAR->LINK = TEMP
            (ii)    set REAR = TEMP
    8.  end if condition
    9.  Exit


DEQUEUE ( ITEM )
    1.  if FRONT = NULL
            (i)     print "UNDERFLOW"
            (ii)    return
    2.  end if condition
    3.  if FRONT = REAR
            (i)     set FRONT = REAR
            (ii)    set REAR = NULL
    4.  else
            (i)     set FRONT = FRONT->LINK
    5. end if condition
    6. Exit
```

PROGRAM

```
#include<iostream>
using namespace std;

class Node
{
 public:
    int data;
    Node *next;
};

class Queue
{
    public:
    Node *front,*rear;
    Queue()
    {front=rear=NULL;
    }

    void enqueue(int item);
    void dequeue();
    void display();
    ~Queue();
};

void Queue::enqueue(int item)
{
    Node *temp=new Node;
    if(temp==NULL)
    {
        cout<<"OVERFLOW"<<endl;
        return;
    }
    temp->data=item;
    temp->next=NULL;

    if(front==NULL)
    {
        front=rear=temp;
    }
    else{
        rear->next=temp;
        rear=temp;
    }
    cout<<"\t"<<item<<" en-queued"<<endl;
}

void Queue::display()
{
    if(front==NULL)
    {
        cout<<"\tUNDERFLOW"<<endl;
        return;
    }
    Node *temp=front;

    while(temp)
    {
        cout<<"\t"<<temp->data<<" -> ";
        temp=temp->next;
    }
    cout<<"NULL";
    cout<<endl;
}

void Queue :: dequeue()
    {
    if (front==NULL)
    {
        cout<<"\tUNDERFLOW\n";
        return;
    }

    cout<<"\t"<<"Element : "<<front->data<<" de-queued "<<endl;
    if(front==rear)
        front=rear=NULL;
    else
        front=front->next;
}

Queue ::~Queue()
```

```
{
    while(front!=NULL)
    {
        Node *temp=front;
        front=front->next;
        delete temp;
    }
    rear=NULL;
}



int main()
{
    Queue Q;
    int value, option,ch;
  cout<<"\n\tQUEUE IMPLEMENTATION USING LINKEDLIST\n";
  do
  {

    cout << "\nMake your choice : \n1. Enqueue()";
    cout << "\n2. Dequeue()";
    cout << "\n3. Display() ";
    cout<<"\n4. Exit";
    cout<<"\n ==> ";

    cin >> option;

    switch (option)
    {
    case 1:
     {
      cout << "------------------------------------------------"//
                            "---------------\n";
      cout << "\tENQUEUE\n";
      cout<<"\nEnter the element insert in Queue: ";
      cin >> value;
       Q.enqueue(value);
      break;
    }
    case 2:
    {
     cout << "---------------------------------------------"//
                            "------------------\n";
      cout << "\tDEQUEUE\n" << endl;
      Q.dequeue();
      break;
    }
    case 3:
     {
      cout << "----------------------------------------------"//
                            "-----------------\n";
      cout << "\tDISPLAY ALL ELEMENTS IN QUEUE\n" << endl;
       Q.display();
      break;
     }
     case 4:
     {
       break;
     }
     default:
      cout << "Wrong option" << endl;
    }
    cout << "---------------------------------------------------"//
                    "------------\n";
    if( option != 4)
    {
      cout<<"Do you want to continue -1 or Exit - 0 : ";
      cin>>ch;
      if(ch == 0)
      {
       option = 4;
      }
    }

  } while (option != 4);
cout<<"\tEND\n";
  return 0;
}
```

SAMPLE INPUT-OUTPUT



```
        QUEUE IMPLEMENTATION USING LINKEDLIST

Make your choice :
1. Enqueue()
2. Dequeue()
3. Display()
4. Exit
 ==> 1
---------------------------------------------------------------
        ENQUEUE

Enter the element insert in Queue: 23
        23 en-queued
---------------------------------------------------------------
Do you want to continue -1 or Exit - 0 : 1

Make your choice :
1. Enqueue()
2. Dequeue()
3. Display()
4. Exit
 ==> 1
---------------------------------------------------------------
        ENQUEUE

Enter the element insert in Queue: 34
        34 en-queued
---------------------------------------------------------------
Do you want to continue -1 or Exit - 0 : 1

Make your choice :
1. Enqueue()
2. Dequeue()
3. Display()
4. Exit
 ==> 3
---------------------------------------------------------------
        DISPLAY ALL ELEMENTS IN QUEUE

        23 ->   34 -> NULL
---------------------------------------------------------------
Do you want to continue -1 or Exit - 0 : 1

Make your choice :
1. Enqueue()
2. Dequeue()
3. Display()
4. Exit
 ==> 2
---------------------------------------------------------------
        DEQUEUE

        Element : 23 de-queued
---------------------------------------------------------------
Do you want to continue -1 or Exit - 0 : 1

Make your choice :
1. Enqueue()
2. Dequeue()
3. Display()
4. Exit
 ==> 4
---------------------------------------------------------------
        END
```