TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
THAPATHALI CAMPUS

LAB DOCUMENTATION
OF
DATA STRUCTURE AND ALGORITHM

Submitted By:
Niraj Duwal (075/BEI/028)

Submitted To:
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
THAPATHALI CAMPUS
KATHMANDU, NEPAL
December 13 , 2020

TABLE OF CONTENTS

LAB 1. Stack

1.1. WAP for array implementation of stack

Problem analysis:

Here, we need to write a program for implementing stack data structure and perform different operations like push, pull and peek on it.

Algorithm:

```
Push:
        Step 1: If TOP = MAX – 1
                    Print “Overflow”
                    Goto step 4
            [End of if]
        Step 2: Set Top = Top + 1
        Step 3: Set Stack [Top] = value
        Step 4: End
Pop:
        Step 1: If Top=Null
                    Print “underflow”
                    Goto step 4
            [End of if]
        Step 2: Set value = Stack [top]
        Step 3: Set Top = Top – 1
        Step 4: End
Peek:
        Step 1: If Top = Null
                    Print “stack is empty”
                    Goto step 3
        Step 2: Return Stack[Top]
        Step 3: End
```

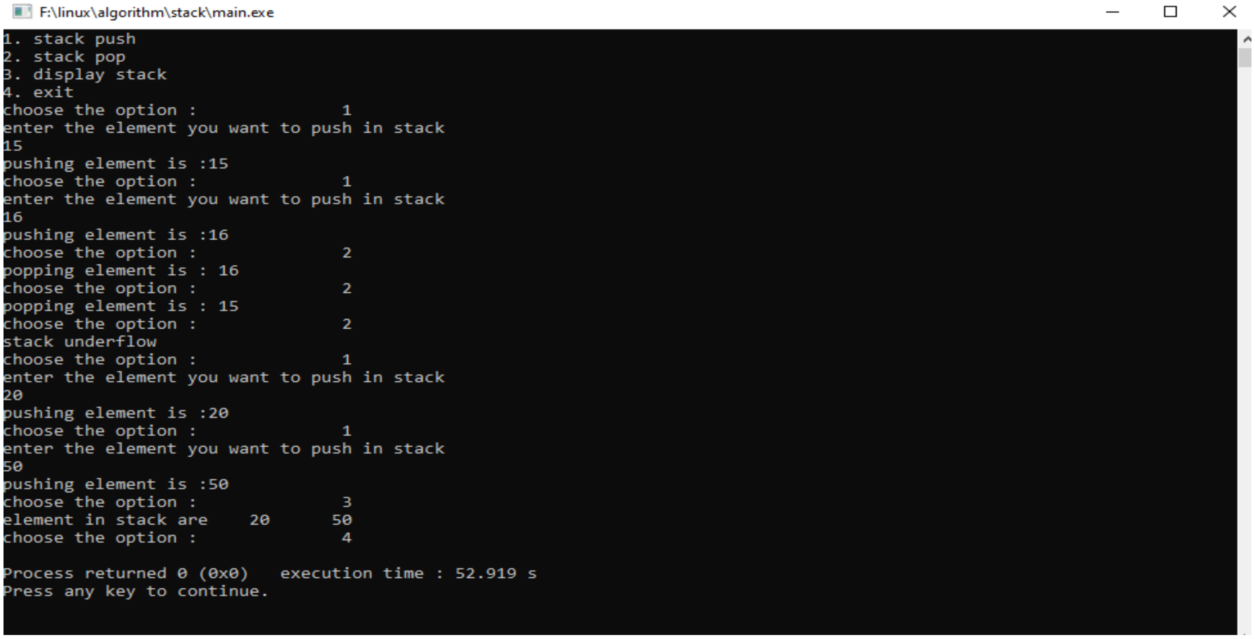
Source Code:

//1.WAP for array implementation of Stack

```
#include <iostream>
using namespace std;
#define max 5
int array[max],stack=-1;
void push()
{
   int val;
   if (stack==max-1)
   {
      cout<<"stack overflow"<<endl;
   }
   else
   {
      cout<<"enter the element you want to push in stack "<<endl;
      cin>>val;
      stack++;
     array[stack]=val;
     cout<<"pushing element is :"<<array[stack]<<endl;
   }
}
void pop()
{
   if(stack==-1)
   {
      cout<<"stack underflow"<<endl;
   }
```

```
    else
    {
        cout<<"popping element is : "<<array[stack]<<endl;
        stack--;
    }
}
void display()
{
    if(stack==-1)
    {
    cout<<"empty stack"<<endl;
    }
    else {
            cout<<"element in stack are \t";
            for(int i=0;i<=stack;i++)
            {
                        cout<<array[i];
                        cout<<"\t";
            }
            }
    cout<<endl;
}
int main (){
    int a;
    cout<<"1. stack push "<<endl;
    cout<<"2. stack pop "<<endl;
    cout<<"3. display stack "<<endl;
    cout<<"4. exit"<<endl;
    do{
    cout<<"choose the option : \t \t ";
    cin>>a;
    switch(a)
    {
      case 1:push();
          break;
      case 2:pop();
          break;
      case 3:display();
          break;
      case 4:break;
      default: cout<<"invalid input"<<endl;
    }
    }while(a!=4);
  }
```

Output :



F:\linux\algorithm\stack\main.exe

```
1. stack push
2. stack pop
3. display stack
4. exit
choose the option :              1
enter the element you want to push in stack
15
pushing element is :15
choose the option :              1
enter the element you want to push in stack
16
pushing element is :16
choose the option :              2
popping element is : 16
choose the option :              2
popping element is : 15
choose the option :              2
stack underflow
choose the option :              1
enter the element you want to push in stack
20
pushing element is :20
choose the option :              1
enter the element you want to push in stack
50
pushing element is :50
choose the option :              3
element in stack are    20      50
choose the option :              4

Process returned 0 (0x0)   execution time : 52.919 s
Press any key to continue.
```

Conclusion:

In this way, stack data structure was studied and implemented using C++. Various operations like push, pop and peek were performed and tested.

1.2 WAP to reverse a list using stack

Problem analysis:

The problem here is to store items in a stack data structure and reverse the list using stack itself.

Algorithm:

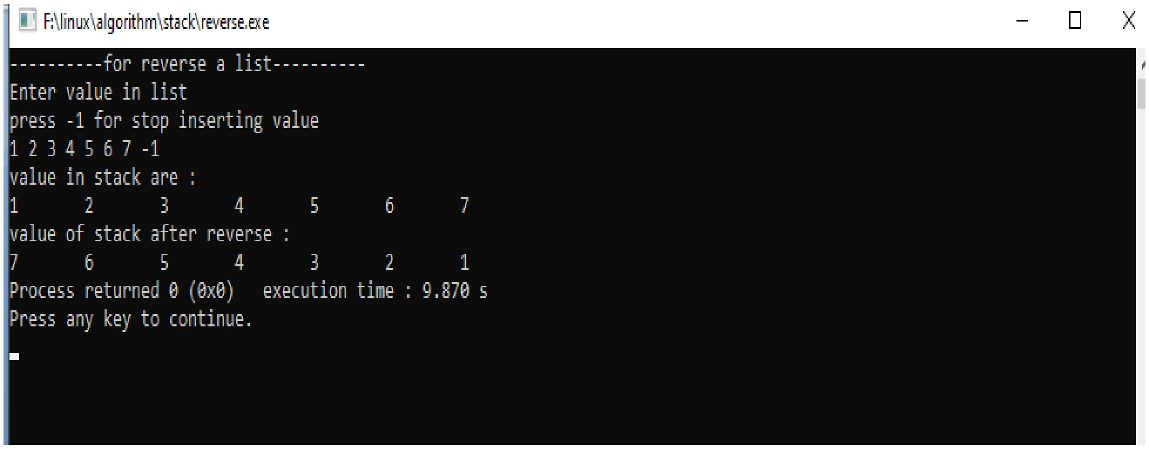
```
Push:
    Step 1: If TOP = MAX – 1
                Print “Overflow”
                Goto step 4
        [End of if]
    Step 2: Set Top = Top + 1
    Step 3: Set Stack [Top] = value
    Step 4: End

Pop:
    Step 1: If Top=Null
                Print “underflow”
                Goto step 4
        [End of if]
    Step 2: Set value = Stack [top]
    Step 3: Set Top = Top – 1
    Step 4: End

Reverse:
    Step 1: Start
    Step 2:  2.1 While Value! = NULL repeat Step 2.2
             2.2 Traverse the list and push all of its nodes onto a stack. I.e. Push(value)
    Step 3:  3.1 While Top! = NULL repeat step 3.2
             3.2 Traverse the list from the head node again and pop a value from the
        stack top and   connect them in reverse order. I.e. Pop ()
    Step 4:  End
```

Source Code:

```
//1.WAP to reverse a list using stack
#include <iostream>
using namespace std;
#define max 10
int top=-1,stack[max];
void push(int x){
      if(top == max-1){
          cout<<"stack overflow"<<endl;
      }
      else {
          top++;
          stack[top]=x;
      }
}
void pop(){
    if(top==-1)
    {
        cout<<"stack underflow"<<endl;
    }
      cout<<stack[top]<<"\t";
      top--;
}
int main()
{
    int val,i;
   cout<<"----------for reverse a list----------"<<endl;
   cout<<"Enter value in list"<<endl;
   cout<<"press -1 for stop inserting value"<<endl;
   cin>>val;
```

```
                do {
                     push(val);
                     cin>>val;
                      }while(val!=-1);
                 cout<<"value in stack are :"<<endl;
                 for(i=0;i<=top;i++)
                 {
                     cout<<stack[i]<<"\t";
                 }
                 cout<<endl;
                 cout<<"value of stack after reverse :"<<endl;
                 while(top!=-1)
                   {
                     pop();
                   }
             }
```

Output :



```
----------for reverse a list----------
Enter value in list
press -1 for stop inserting value
1 2 3 4 5 6 7 -1
value in stack are :
1       2       3       4       5       6       7
value of stack after reverse :
7       6       5       4       3       2       1
Process returned 0 (0x0)   execution time : 9.870 s
Press any key to continue.
```

Conclusion:

In this way by using stack we can reverse a list. Various operations like push and pop were performed and tested. Using stack, reverse of a list was obtained by using pop operation until the stack is empty.

1.3 WAP to check parenthesis of algebraic expression using stack

Problem analysis:

The problem here is to check parenthesis of the algebraic expression given by user is balanced or not by using stack. If the opening and closing parenthesis are equal in numbers, they are balanced

Algorithm :

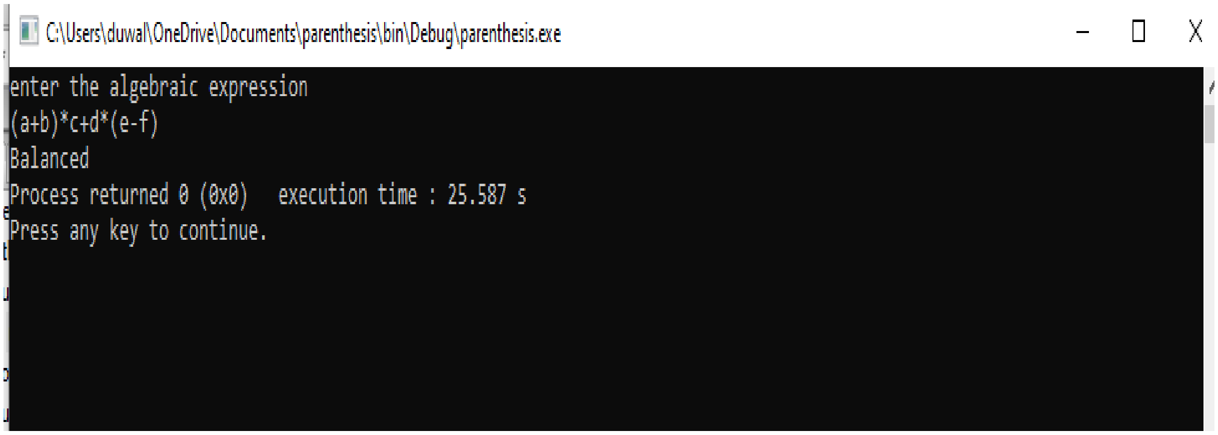
```
Push:
    Step 1: If TOP = MAX – 1
                Print “Overflow”
                Goto step 4
        [End of if]
    Step 2: Set Top = Top + 1
    Step 3: Set Stack [Top] = value
    Step 4: End
Pop:
    Step 1: If Top=Null
                Print “underflow”
                Goto step 4
        [End of if]
    Step 2: Set value = Stack [top]
    Step 3: Set Top = Top – 1
    Step 4: End

Check Parenthesis:
    Step 1: Start
    Step 2: Declare character stack s, string exp
    Step 3:  Traverse the expression string exp
            If  exp = ‘(‘
                Push it to stack
            Else if exp = ‘)’
                Pop character
    Step 4 :  After complete traversal,
            If stack=NULL
                Print ‘balanced’
            Else
                Print ‘not balanced’
    Step 5: End
```

Source Code :

```
#include <iostream>
#define SIZE 5
using namespace std;
class stack
{
   private:
      int tos=-1;
      int array[SIZE];
   public:
      bool isEmpty()
      {
         if(tos==-1)
         {
            return true;
         }
         else
         {
            return false;
         }
      }
      bool isFull()
      {
         if(tos==(SIZE-1))
```

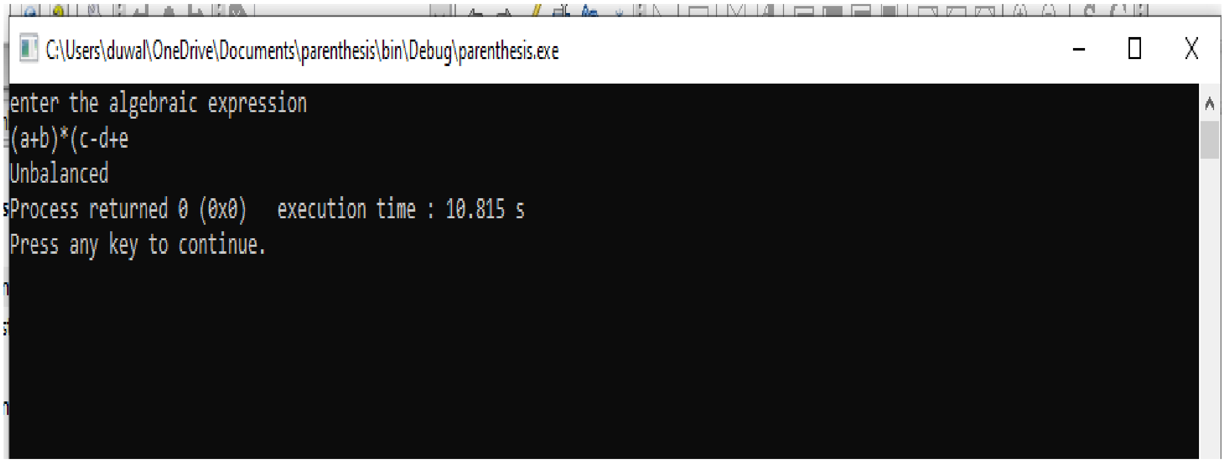
```
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        void push(int value)
        {
            if(isFull()==true)
                cout<<"Stack Overflow";
            else
            {
                tos++;
                array[tos] = value;
            }
        }
        void pop()
        {
            if(isEmpty()==true)
                cout<<"Stack underflow";
            else
            {
                tos--;
            }
        }
};
bool check_parenthesis(string infix)
{
    stack stk;
    for(int i=0; i<infix.length(); i++)
    {
        if(infix[i]=='(')
            stk.push('(');
        else if(infix[i]==')')
        {
            if(stk.isEmpty()!=true)
                stk.pop();
            else
                return false;
        }
    }
    if(stk.isEmpty()==true)
        return true;
    else if(stk.isEmpty()==false)
        return false;
}
int main()
{
    string infix_exp;
    cout<<"enter the algebraic expression "<<endl;
    cin>>infix_exp;
    bool check = check_parenthesis(infix_exp);
    if(check==true)
        cout<<"Balanced";
    else
        cout<<"Unbalanced";
}
```

Output :



C:\Users\duwal\OneDrive\Documents\parenthesis\bin\Debug\parenthesis.exe

```
enter the algebraic expression
(a+b)*c+d*(e-f)
Balanced
Process returned 0 (0x0)   execution time : 25.587 s
Press any key to continue.
```



C:\Users\duwal\OneDrive\Documents\parenthesis\bin\Debug\parenthesis.exe

```
enter the algebraic expression
(a+b)*(c-d+e
Unbalanced
Process returned 0 (0x0)   execution time : 10.815 s
Press any key to continue.
```

Conclusion :

In this way by using stack we can check whether the parenthesis of algebraic expression is balanced or not .

1.4 WAP to convert infix to postfix using Stack

Problem analysis :

The problem here is to convert the infix given by the user to postfix by using stack.

Algorithm :

Let, X is an arithmetic expression written in infix notation. This algorithm finds the equivalent postfix expression Y.

```
Step 1 :   Start
Step 2 :   Push “(“onto Stack, and add “)” to the end of X.
Step 3 :   Scan X from left to right and repeat Step 4 to 7 for each element of X until the Stack is
           empty.
Step 4 :  If an operand is encountered, add it to Y.
Step 5 :  If a left parenthesis is encountered, push it onto Stack.
Step 6 :  If an operator is encountered ,then:
                    6.1 Repeatedly pop from Stack and add to Y each operator (on the top of Stack)
                    which    has the same precedence as or higher precedence than operator.
                    6.2 Add operator to Stack.
          [End of If]
Step 7 :  If a right parenthesis is encountered ,then:
                    7.1 Repeatedly pop from Stack and add to Y each operator (on the top of Stack)
                    until a left parenthesis is encountered.
                    7.2 Remove the left Parenthesis.
          [End of If]
Step 8 :  END.
```

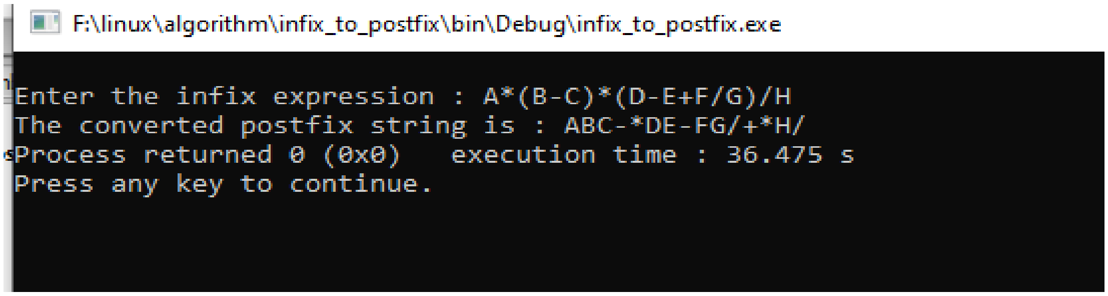
Source Code :

```
#include <iostream>

#define max 10

using namespace std;

char stack[max];

int top=-1;
```

```
void push(char a)
{
    if(top==max-1)
    {
        cout<<"stack overflow"<<endl;
    }
    else
    {
        top++;
        stack[top]=a;
    }
}
char pop()
{
    char c;
    if(top==-1)
    {
        cout<<"stack underflow"<<endl;
    }
    else{
        c=stack[top];
        stack[top]='\0';
        top--;
        return c;
    }
}
int priority(char p)
{
    if(p=='$')
    {
    return 3;
    }
    if ((p=='*')||(p=='/'))
    {
    return 2;
    }
    if ((p=='+')||(p=='-'))
    {
    return 1;
    }
    return 0;
}
string Convert(string infix)
{
    int i=0;
    string postfix ="";
    while(infix[i]!='\0')
    {
        if(infix[i]>='a' && infix[i]<='z'|| infix[i]>='A'&& infix[i]<='Z')
        {
            postfix.insert(postfix.end(),infix[i]);
            i++;
        }
        else if(infix[i]=='(' || infix[i]=='{'  || infix[i]=='[')
        {
            push(infix[i]);
            i++;
        }
        else if(infix[i]==')' || infix[i]=='}'  || infix[i]==']')
        {
            if(infix[i]==')')
            {
                while(stack[top]!='(')
                {
                 postfix.insert(postfix.end(),pop());
                }
                pop();
                i++;
            }
            if(infix[i]==']')
            {
                while(stack[top]!='[')
                {
                    postfix.insert(postfix.end(),pop());
                }
                pop();
                i++;
            }
            if(infix[i]=='}')
            {
                while(stack[top]!='{')
                {
                    postfix.insert(postfix.end(),pop());
                }
                pop();
                i++;
            }
        }
        else
        {
            if(top==-1)
            {
                push(infix[i]);
                i++;
            }
            else if( priority(infix[i]) <= priority(stack[top])) {
                postfix.insert(postfix.end(),pop());

                while(priority(stack[top]) == priority(infix[i])){
                    postfix.insert(postfix.end(),pop());
                    if(top < 0) {
                        break;
                    }
                }
                push(infix[i]);
                i++;
            }
            else if(priority(infix[i]) > priority(stack[top])) {
                push(infix[i]);
                i++;
            }
        }
```

```
    }
    while(top!=-1)
    {
        postfix.insert(postfix.end(),pop());
    }
    cout<<"The converted postfix string is : "<<postfix; //it will print postfix conversion
    return postfix;
}
int main()
{
    int cont;
    string infix, postfix;
    cout<<"\nEnter the infix expression : "; //enter the expression
    cin>>infix;
    postfix = Convert(infix);
    return 0;
}
```

Output:



F:\linux\algorithm\infix_to_postfix\bin\Debug\infix_to_postfix.exe

```
Enter the infix expression : A*(B-C)*(D-E+F/G)/H
The converted postfix string is : ABC-*DE-FG/+*H/
Process returned 0 (0x0)   execution time : 36.475 s
Press any key to continue.
```

Conclusion :

In this program, we have to manage the operator according to its priority order. According to its priority we can push that operator to stack and then pushed it to postfix. Operand is directly pushed to stack. In this way by using above algorithm and source code we can convert infix to postfix by using stack.

1.5 WAP to evaluate postfix expression using Stack

Problem analysis:

In this program we evaluate the postfix expression using stack. The following algorithm, which uses STACK to hold operands, evaluates the post fix expression.

Algorithm :

```
Step 1 :  Add ")" at the end of the postfix expression
Step 2 :  Scan every character of the postfix expression and repeat steps 3 and 4
          until ")" is encountered
Step 3 : IF an operand is encountered,  push it on the stack
              IF an operator 0 is encountered, then
                 a. Pop the two elements from the stack as A and B
                 b. Evaluate B 0 A where A is the topmost element and B is the element below A
                 c. Push the result of evaluation on the stack
         [END OF IF]
Step 4 :  Set result equal to the topmost element of the stack
Step 5 :  Exit
```

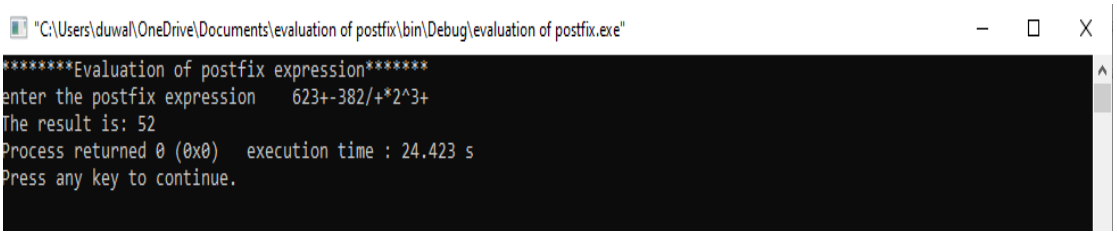
Sourcecode:

```
#include<iostream>
#include<cmath>
#include<stack>
using namespace std;
float scanNum(char ch) {
  int value;
  value = ch;
  return float(value-'0');   //return float from character
}
int isOperator(char ch) {
  if(ch == '+'|| ch == '-'|| ch == '*'|| ch == '/' || ch == '^')
    return 1;    //character is an operator
  return -1;   //not an operator
```

```
}
int isOperand(char ch) {
   if(ch >= '0' && ch <= '9')
      return 1;   //character is an operand
   return -1;   //not an operand
}
float operation(int a, int b, char op) {
   //Perform operation
   if(op == '+')
      return b+a;
   else if(op == '-')
      return b-a;
   else if(op == '*')
      return b*a;
   else if(op == '/')
      return b/a;
   else if(op == '^')
      return pow(b,a); //find b^a
   else
      return INT_MIN; //return negative infinity
}
float postfixEval(string postfix) {
   int a, b;
   stack<float> stk;
   string::iterator it;
   for(it=postfix.begin(); it!=postfix.end(); it++) {
      //read elements and perform postfix evaluation
      if(isOperator(*it) != -1) {
```

```
      a = stk.top();
      stk.pop();
      b = stk.top();
      stk.pop();
      stk.push(operation(a, b, *it));
    }else if(isOperand(*it) > 0) {
      stk.push(scanNum(*it));
    }
  }
  return stk.top();
}
main() {
  string post;
  cout<<"********Evaluation of postfix expression*******"<<endl;
  cout<<"enter the postfix expression \t";
  cin>>post;
  cout << "The result is: "<<postfixEval(post);
}
```

Output:


"C:\Users\duwal\OneDrive\Documents\evaluation of postfix\bin\Debug\evaluation of postfix.exe"

```
********Evaluation of postfix expression*******
enter the postfix expression    623+-382/+*2^3+
The result is: 52
Process returned 0 (0x0)   execution time : 24.423 s
Press any key to continue.
```

Conclusion:

In this way postfix expression is evaluated using stack .

1.6 WAP to convert infix expression into prefix

Problem analysis :

The problem here is to convert an infix expression into prefix expression. To do so, we first convert the infix expression to postfix expression and reverse the converted expression to get the prefix expression. We use stack in this problem as it requires push and pop operation to convert infix expression to postfix and reversing it to gain prefix expression.

ALGORITHM

```
Step 1:Check if the stack is empty
        If stack.top = NULL:
        Print “Stack is empty”
Step 2:Traverse through the expression and evaluate postfix expression
Step 3:While char != NULL:
        If char = ‘(’:
        Replace with ‘)’
        Else if char = ‘)’:
        Replace with ‘(’
Step 4:Reverse(postfix.begin(), postfix.end())
        Print result
Step 5:EXIT
```

Sourcecode:

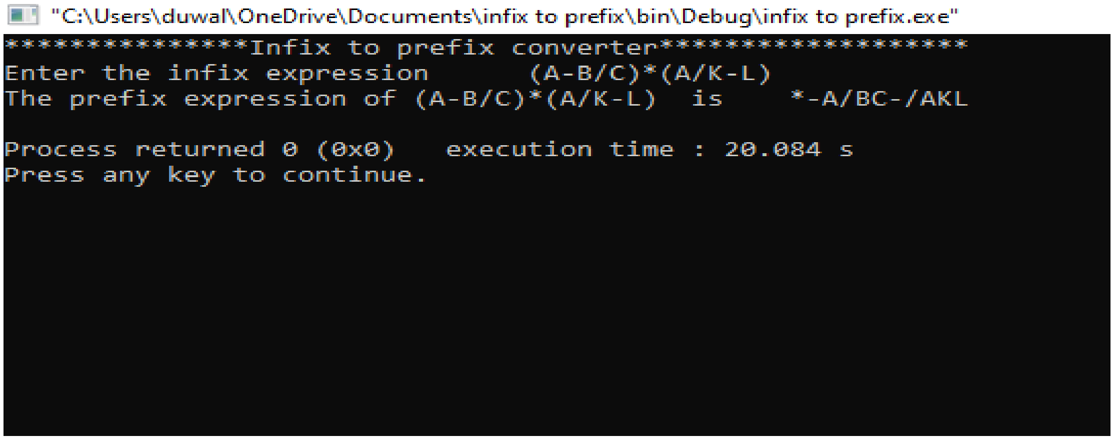
```
#include <iostream>
#include <stack>
#include <algorithm>
using namespace std;
// Function to get the precedence of operators
int precedence(char op) {
if(op == '^')
return 3;
else if (op == '*' || op == '/')
```

```
return 2;
else if (op == '+' || op == '-')
return 1;
else
return -1;
}
// function to convert infix to postfix (returns a string)
string infixToPostfix(string infix) {

// add '(' and ')' at the start & end of infix respectively
    infix = '(' + infix + ')';
    int len = infix.length();
    stack <char> st;
    string postfix;
   // loop through the infix string
    for(int i = 0; i < len; i++) {
if((infix[i] >= 'a' && infix[i] <= 'z') || (infix[i] >= 'A' && infix[i] <= 'Z')) {
            postfix += infix[i];
        }
       else if (infix[i] == '(')
            st.push('(');
        else if (infix[i] == ')') {
            while(st.top() != '(') {
                postfix += st.top();
                st.pop();
            }
            // remove '(' from the stack
            st.pop();
```

```
                }
                else {
                        // Operator is found in scanned char
                        while(precedence(infix[i]) <= precedence(st.top())) {
                                postfix += st.top();
                                st.pop();
                        }
                        st.push(infix[i]);
                }
        }
                return postfix;
}
// Infix to prefix
string infixToPrefix(string infix) {
int l = infix.length();
// Reverse the infix expression
reverse(infix.begin(), infix.end());
// Replace ( with ) and vice versa to the reversed string
        for(int i = 0; i < l; i++) {
                if(infix[i] == '(') {
                        infix[i] = ')';
                        i++;
                }
                else if (infix[i] == ')') {infix[i] = '(';    i++;}}
string prefix = infixToPostfix(infix);
// reversing the postfix expression
reverse(prefix.begin(), prefix.end());
return prefix;
```

```
}
int main() {
   string infix;
   cout<<"***************Infix to prefix converter*******************"<<endl;
   cout<<"Enter the infix expression \t";
   cin>>infix;
   cout<<"The prefix expression of "<<infix<<"  is\t";
         cout << infixToPrefix(infix) << endl;
         return 0;
}
```

Output :



"C:\Users\duwal\OneDrive\Documents\infix to prefix\bin\Debug\infix to prefix.exe"

```
***************Infix to prefix converter*******************
Enter the infix expression      (A-B/C)*(A/K-L)
The prefix expression of (A-B/C)*(A/K-L)  is    *-A/BC-/AKL

Process returned 0 (0x0)   execution time : 20.084 s
Press any key to continue.
```

Conclusion :

The infix expression is first reversed then convert the reversed infix to postfix then reversed the output postfix expression and finally infix expression is converted to prefix expression. Thus the program for evaluation of postfix expression using stack was evaluated successfully.

1.7 WAP to evaluate a prefix expression

Problem analysis:

We input the prefix expression. It’s characters are scanned and proper operation is done as per those charcters. Finally, the result of prefix expression is popped and displayed.

Algorithm :

```
Step 1 : Scan every character from right to left .
Step 2 : If the element is an operand, push it into the stack.
Step 3 : If the element is an operator O, pop twice and two operands.
         Calculate and push it  back to the stack.
Step 4 : When the expression is ended, the value in the stack is the final
           answer.
```

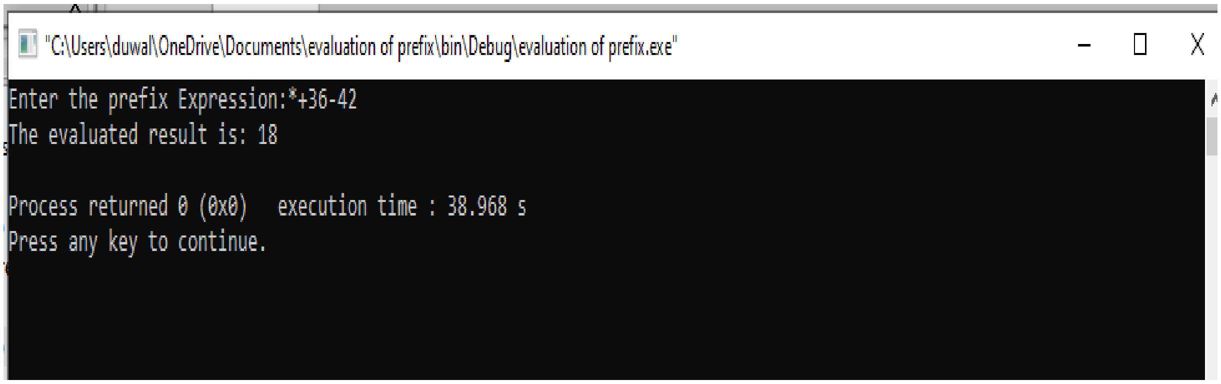
Sourcecode:

```
#include<iostream>
#include<math.h>
#define max 100
using namespace std;
class stack
{
   int data[max];
   int top;
public:
   stack()
   {
      top=-1;
   }
   int isFull()
   {
      if(top==(max-1))
```

```
        return(1);
    return(0);
}
int isEmpty()
{
    if(top==-1)
    {
        return(1);
    }
    else
    {
        return 0;
    }
}
void push(int val)
{
    int i;
    i=isFull();
    if(i==0)
    {
        top++;
        data[top]=val;
    }
}
int pop()
{
    int i;
    int value;
```

```
        i=isEmpty();
        if(i==0)
        {
            value=data[top];
            top--;
        }
        return value;
    }
};
float prefix_evaluate(stack s,string postfix)
{
    int a,b,x;
    float result;
    for(int i=postfix.length()-1;i>=0;i--)
    {
        if(postfix[i]>='0' && postfix[i]<='9')
        {
            x=postfix[i]-48;
            s.push(x);
        }
        else if(postfix[i]=='+'|| postfix[i]=='-'|| postfix[i]=='*'|| postfix[i]=='/'|| postfix[i]=='$')
        {
            a=s.pop();
            b=s.pop();
            switch(postfix[i])
            {
            case'+':
                result=a+b;
                s.push(result);
                break;
            case'-':
                result=a-b;
                s.push(result);
                break;
             case'*':
                result=a*b;
                s.push(result);
                break;
            case'/':
                result=a/b;
                s.push(result);
                break;
            case'$':
                result=pow(b,a);
                s.push(result);
                break;
            }
        }
    }
    return s.pop();
}
int main()
{
    stack s;
    string prefix_exp;
    float evaluated_result;
    cout<<"Enter the prefix Expression:";
    cin>>prefix_exp;
    evaluated_result=prefix_evaluate(s,prefix_exp);
    cout<<"The evaluated result is: "<<evaluated_result<<endl;
    return 0;
}
```

Output:



"C:\Users\duwal\OneDrive\Documents\evaluation of prefix\bin\Debug\evaluation of prefix.exe"

```
Enter the prefix Expression:*+36-42
The evaluated result is: 18

Process returned 0 (0x0)   execution time : 38.968 s
Press any key to continue.
```

Conclusion :

In this way the prefix expression is evaluated using stack.

LAB 2. Queue

2.1 WAP for array implementation of Linear queue

Problem analysis:
Here, we need to write a program for implementing linear queue data structure and perform different operations like enqueue and dequeue on it.

Algorithm :

```
Enqueue:
        Step 1. If rear = max – 1
                   Write ‘overflow’
                   Goto step 4
                [end of if]
        Step 2. If front = -1 and rear = -1
                   Set front = rear = 0
                Else
                   Set rear = rear + 1
                [end of if]
        Step 3. Set queue[rear] = val
        Step 4. Exit
Dequeue:
        Step 1: If front = -1 or front > rear
                   Write ‘underflow’
                Else
                   Set val = queue[front]
                   Set front = front + 1
                [end of if]
        Step 2: Exit
Display:
        Step 1: If rear<front or front = -1
                    Write ‘empty’
                    Goto Step 3
        Step 2:  Return queue
        Step 3:  Exit
```

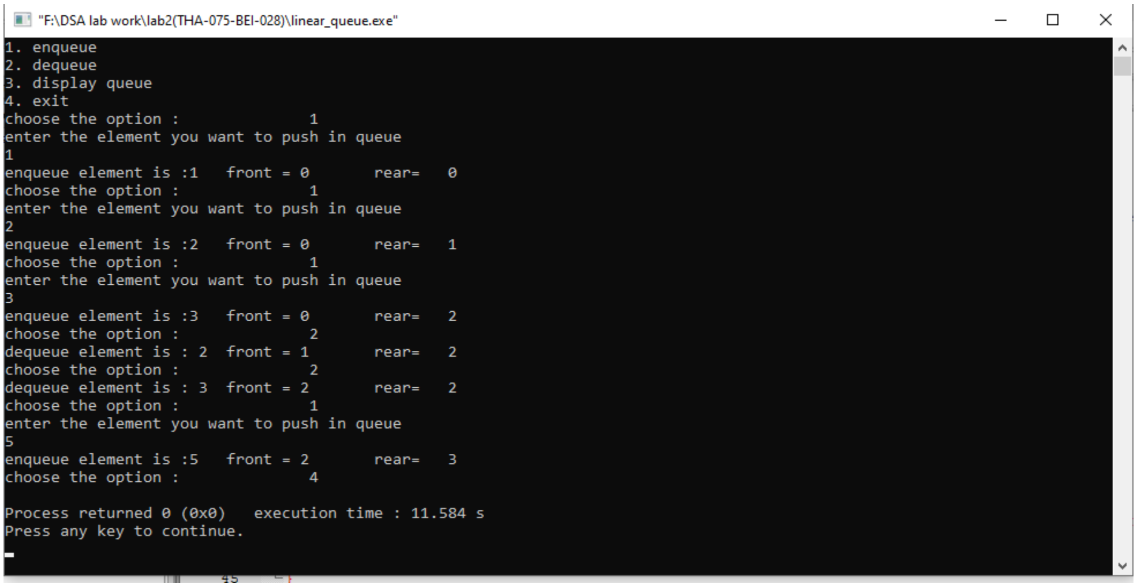
Source Code:

```
#include <iostream>

#define max 5

using namespace std;

int array[max],front=-1,rear=-1;

void enqueue()
```

```
{
    int val;
    if (rear>=max-1)
    {
        cout<<"Queue overflow"<<endl;
    }
    else
    {
        if(front==-1 && rear==-1)
        {
            rear=0;
            front=0;
        }
        else
        {
        rear++;
        }
        cout<<"enter the element you want to push in queue"<<endl;
        cin>>val;
       array[rear]=val;
       cout<<"enqueue element is :"<<array[rear]<<"\t"<<"front =\t"<<front<<"\trear=
\t"<<rear<<endl;
    }
}
void dequeue()
{
    if(rear<front || front==-1)
    {
        cout<<"queue underflow"<<endl;
    }
    else
    {
        array[front]=0;
        front++;
        cout<<"dequeue element is : "<<array[front]<<"\t"<<"front =\t"<<front<<"\trear=
\t"<<rear<<endl;
    }
}
void display()
{
    if(rear<front || front==-1)
    {
    cout<<"empty queue"<<endl;
    }
    else {
    cout<<"element in queue are \t";
    for(int i=0;i<=max-1;i++)
    {
        cout<<array[i];
        cout<<"\t";
    }
    cout<<"front =\t"<<front<<"\trear= \t"<<rear<<endl;
    }
    cout<<endl;
}
int main (){
    int a;
    cout<<"1. enqueue "<<endl;
    cout<<"2. dequeue "<<endl;
    cout<<"3. display queue "<<endl;
    cout<<"4. exit"<<endl;
    do{
```

```
    cout<<"choose the option : \t \t ";
    cin>>a;
    switch(a)
    {
        case 1:enqueue();
                break;
        case 2:dequeue();
                break;
        case 3:display();
                break;
        case 4:break;
        default: cout<<"invalid input"<<endl;
    }
    }while(a!=4);
}
```

Output:



```
1. enqueue
2. dequeue
3. display queue
4. exit
choose the option :             1
enter the element you want to push in queue
1
enqueue element is :1   front = 0       rear=   0
choose the option :             1
enter the element you want to push in queue
2
enqueue element is :2   front = 0       rear=   1
choose the option :             1
enter the element you want to push in queue
3
enqueue element is :3   front = 0       rear=   2
choose the option :             2
dequeue element is : 2  front = 1       rear=   2
choose the option :             2
dequeue element is : 3  front = 2       rear=   2
choose the option :             1
enter the element you want to push in queue
5
enqueue element is :5   front = 2       rear=   3
choose the option :             4

Process returned 0 (0x0)   execution time : 11.584 s
Press any key to continue.
```

Conclusion:

In this way, linear queue data structure was studied with its operation enqueue and dequeue.

2.2 WAP for implementation of circular queue

Problem analysis:

Here, we need to write a program for implementing circular queue data structure and perform different operations like enqueue and dequeue on it.

Algorithm:

Inserting an element:

```
Step 1. If front = 0 and rear = max – 1 or front = rear + 1
            Write “overflow”
            Goto step 4
Step 2. If front = -1 and rear = -1
            Set front = rear = 0
        Else if rear = max – 1 and front != 0
            Set rear = 0
        Else
            Set rear = rear + 1
        [end of if]
Step 3. Set queue[rear] = val
Step 4. Exit
```

Deleting an element:

```
Step 1. If front = -1
            Write “underflow”
            Goto step 4
        [end of if]
Step 2. Set val = queue[front]
Step 3. If front = rear
            Set front = rear = -1
        Else
            If front = max – 1
                Set front = 0
            Else
                Set front = front + 1
            [end of if]
        [end of if]
Step 4. Exit
```

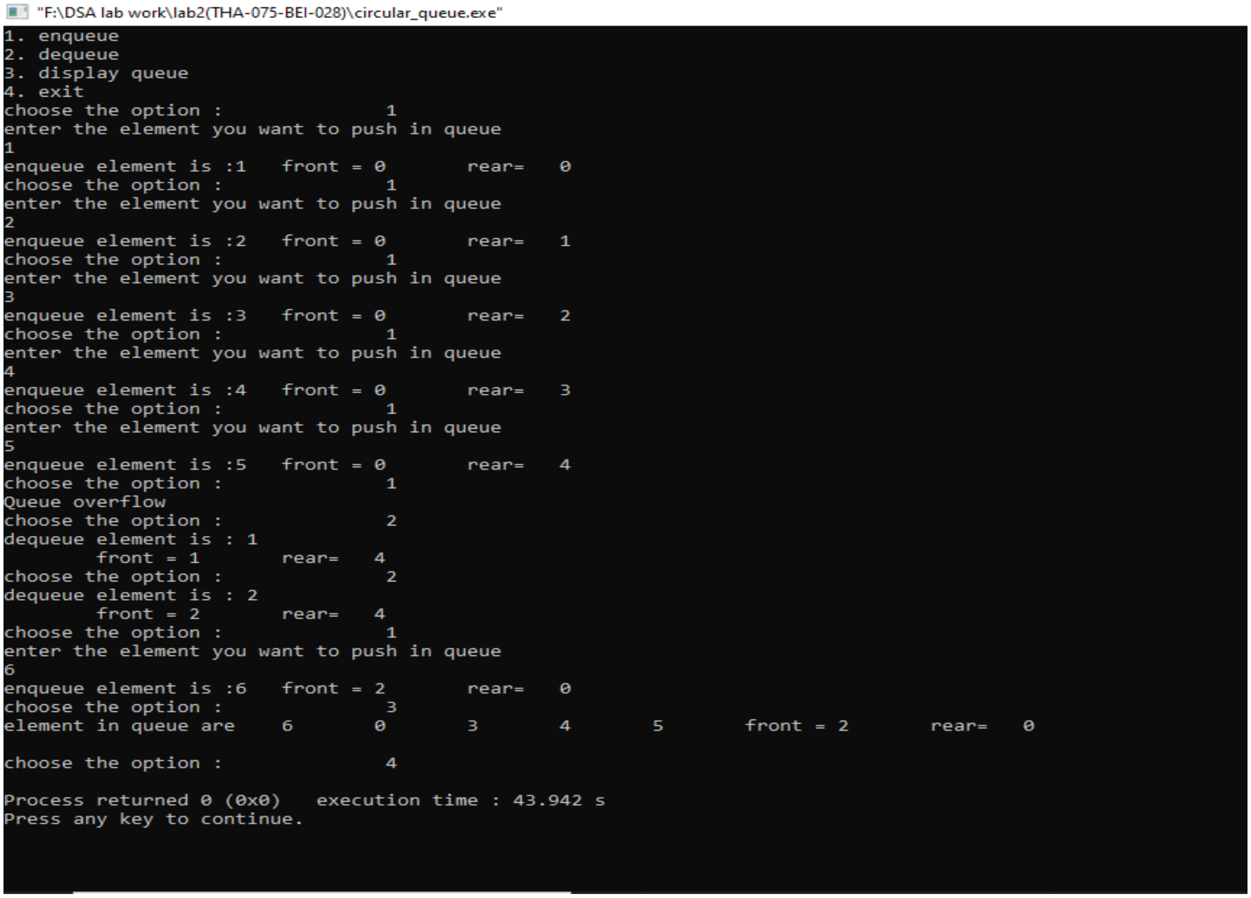
Display :

```
Step 1: If rear=-1 and front=-1
            Write ‘empty’
            Goto step 3
Step 2: Return queue
Step 3 : Exit
```

Source code:

```
#include <iostream>
#define max 5
using namespace std;
int array[max],front=-1,rear=-1;
void enqueue()
{
    int val;
    if ((front==0 && rear==max-1) || (front==rear+1))
    {
        cout<<"Queue overflow"<<endl;
    }
    else
    {
        if(front==-1 && rear==-1)
        {
            front=0;
            rear=0;
        }
        else if (rear==max-1)
        {
            rear=0;
        }
        else
        {
        rear++;
        }
        cout<<"enter the element you want to push in queue "<<endl;
        cin>>val;
       array[rear]=val;
       cout<<"enqueue element is :"<<array[rear]<<"\t"<<"front =\t"<<front<<"\trear=
\t"<<rear<<endl;
    }

}
void dequeue()
{
    if(rear==-1 && front==-1)
    {
        cout<<"queue underflow"<<endl;
    }
    else
    {
    cout<<"dequeue element is : "<<array[front]<<endl;
    array[front]=0;
        if(front==rear)
        {
           front=-1;
           rear=-1;
        }
        else if (front==max-1)
        {
           front=0;
        }
        else{
           front++;
        }
        cout<<"\t"<<"front =\t"<<front<<"\trear= \t"<<rear<<endl;
    }
}
void display()
{
    if(rear==-1 && front==-1)
    {
    cout<<"empty queue"<<endl;
    }
    else {
    cout<<"element in queue are \t";
    for(int i=0;i<=max-1;i++)
    {
        cout<<array[i];
        cout<<"\t";
    }
    cout<<"front =\t"<<front<<"\trear= \t"<<rear<<endl;
    }
    cout<<endl;
}
int main (){
    int a;
    cout<<"1. enqueue "<<endl;
    cout<<"2. dequeue "<<endl;
    cout<<"3. display queue "<<endl;
    cout<<"4. exit"<<endl;
    do{
    cout<<"choose the option : \t \t ";
    cin>>a;
    switch(a)
    {
        case 1:enqueue();
                break;
        case 2:dequeue();
                break;
        case 3:display();
               break;
        case 4:break;
        default: cout<<"invalid input"<<endl;
    }
    }while(a!=4);

}
```

Output :



```
"F:\DSA lab work\lab2(THA-075-BEI-028)\circular_queue.exe"
1. enqueue
2. dequeue
3. display queue
4. exit
choose the option :              1
enter the element you want to push in queue
1
enqueue element is :1   front = 0       rear=   0
choose the option :              1
enter the element you want to push in queue
2
enqueue element is :2   front = 0       rear=   1
choose the option :              1
enter the element you want to push in queue
3
enqueue element is :3   front = 0       rear=   2
choose the option :              1
enter the element you want to push in queue
4
enqueue element is :4   front = 0       rear=   3
choose the option :              1
enter the element you want to push in queue
5
enqueue element is :5   front = 0       rear=   4
choose the option :              1
Queue overflow
choose the option :              2
dequeue element is : 1
        front = 1       rear=   4
choose the option :              2
dequeue element is : 2
        front = 2       rear=   4
choose the option :              1
enter the element you want to push in queue
6
enqueue element is :6   front = 2       rear=   0
choose the option :              3
element in queue are    6       0       3       4       5       front = 2       rear=   0

choose the option :              4

Process returned 0 (0x0)   execution time : 43.942 s
Press any key to continue.
```

Conclusion :

In this way, circular queue data structure was studied with its operation insertion and deletion.

LAB 3: List

3.1 WAP to implement contiguous list using array

Problem analysis:

List is a collection of nodes. A pointer to a node is represented by array index whose value lies between 0 and max-1. Null pointer is represented by -1. There must be separate function to get the available nodes and free the nodes.

Algorithm:

Getnode:

```
    Step 1. If avail = NULL
            Write overflow
            Goto step 5
    Step 2. Set pointer ptr = avail
    Step 3. Set avail = node[avail].next
    Step 4. Return ptr
    Step 5. Exit
```

Freenode:

```
    Step 1. Input a pointer ptr
    Step 2. Set node[ptr].next = avail
    Step 3. Set avail = ptr
    Step 4. Stop
```

Delete node:

```
    Step 1. Input ptr
    Step 2. node[ptr].info = 0
    Step 3. If ptr = null or ptr > size – 1
            Write “Invalid node”
            Goto step 6
    Step 4. node[ptr-1].next = node[ptr].next
    Step 5. Free node ptr
    Step 6. Exit
Insert after a pointer:
    Step 1. Input a value val and pointer ptr
    Step 2. if ptr = null
        Write “Invalid insertion”
        Goto step 7
    Step 3. newptr = available node
    Step 4. node[newptr].info = val
    Step 5. node[newptr].next = node[ptr].next
    Step 6. node[ptr].next = newptr
    Step 7. Exit
Delete after a pointer:
    Step 1. Input a pointer ptr
    Step 2. If ptr = null or node[ptr].next = null
        Write “Invalid deletion”
        Goto step 7
    Step 3. delptr = node[ptr].next
    Step 4. delval = node[delptr].info
    Step 5. node[ptr].next = node[delptr].next
    Step 6. free node delptr
    Step 7. Exit
```

Source code:

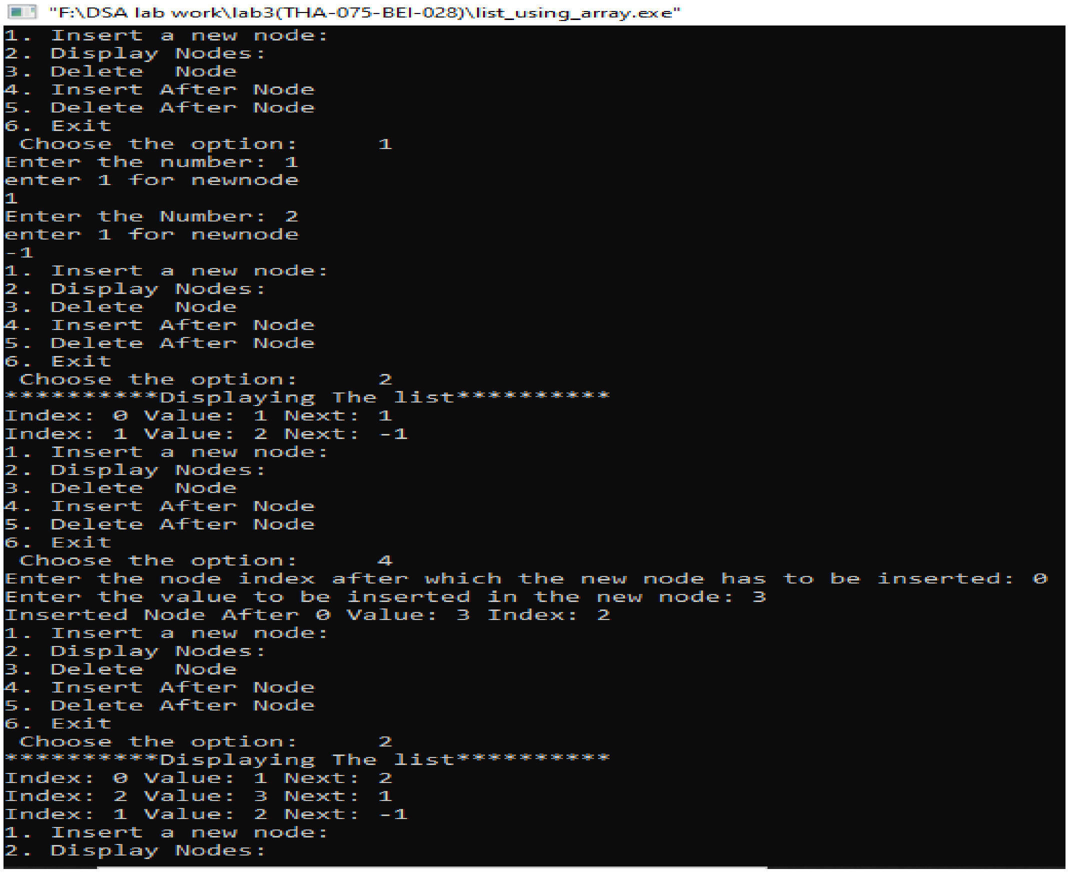
```
//WAP to implement contiguous list using array
#include<iostream>
#define max 15
using namespace std;
struct nodetype
{
    int info,next;
};
class list
{
    struct nodetype node[max];
    int avail=0;
public:
    int intialize_availlist()
    {
        int i;
        for(i=0;i<max-1;i++)
        {
            node[i].next=i+1;
        }
        node[max-1].next=-1;
    }
    int get_node()
    {
        int p;
        if(avail==-1)
        {
            cout<<"Overflow";
        }
        p=avail;
        avail=node[avail].next;
        return p;
    }
    int freenode(int p)
```

```
{
    node[p].next=avail;
    avail=p;
}
void insertnode(int &list1)
{
    int val,ptr,curptr,newnode=1;
    while(newnode==1)
    {
        if(list1==-1)
        {
            ptr=get_node();
            list1=ptr;
            cout<<"Enter the number: ";
            cin>>val;
            node[ptr].info=val;
            node[ptr].next=-1;
        }
        else
        {
            curptr=0;
            while(node[curptr].next!=-1)
            {
                curptr=node[curptr].next;
            }
            ptr=get_node();
            cout<<"Enter the Number: ";
            cin>>val;
            node[curptr].next=ptr;
            node[ptr].info=val;
            node[ptr].next=-1;
        }
        cout<<"enter 1 for newnode"<<endl;
        cin>>newnode;
    }
}
int displaynode()
{
    cout<<"**********Displaying The list**********"<<endl;
    int i;
    int ptr=0;
    if(avail==0)
    {
        cout<<"List Underflow"<<endl;
    }
    while(ptr!=-1)
    {
        cout<<"Index: "<<ptr<<" Value: "<<node[ptr].info<<" Next: "<<node[ptr].next<<endl;
        ptr=node[ptr].next;
    }
}
int deletenode(int &list1)
{
    int val,curptr,preptr=-1;
    curptr = list1;
    while(node[curptr].next!=-1)
    {
        preptr=curptr;
        curptr=node[curptr].next;
    }
    freenode(curptr);
    cout<<endl;
    cout<<"The deleted value is: "<<node[curptr].info<<endl;
    if(preptr==-1)
    {
        list1=-1;
    }
    else
    {
        node[preptr].next=-1;
    }
}
int insert_after(int ptr,int val)
{
    int newptr;
    if(ptr==-1)
    {
        cout<<"Invalid Insertion";
    }
    else
    {
        newptr=get_node();
        node[newptr].info=val;
        node[newptr].next=node[ptr].next;
```

```
            node[ptr].next=newptr;
            cout<<"Inserted Node After "<<ptr<<" Value: "<<val<<" Index: "<<newptr<<endl;
        }
    }
    int delete_after(int ptr)
    {
        int delptr,delval;
        if(ptr==-1 || node[ptr].next==-1)
        {
            cout<<"Invalid deletion after given ptr"<<endl;
        }
        else{
            delptr=node[ptr].next;
            delval=node[delptr].info;
            cout<<"Deleted Value is "<<delval<<endl;
            node[ptr].next=node[delptr].next;
            freenode(delptr);
        }
    }
};
int main()
{
    list l;
    l.intialize_availlist();
    int ch;
    int list1=-1;
    do
    {
```

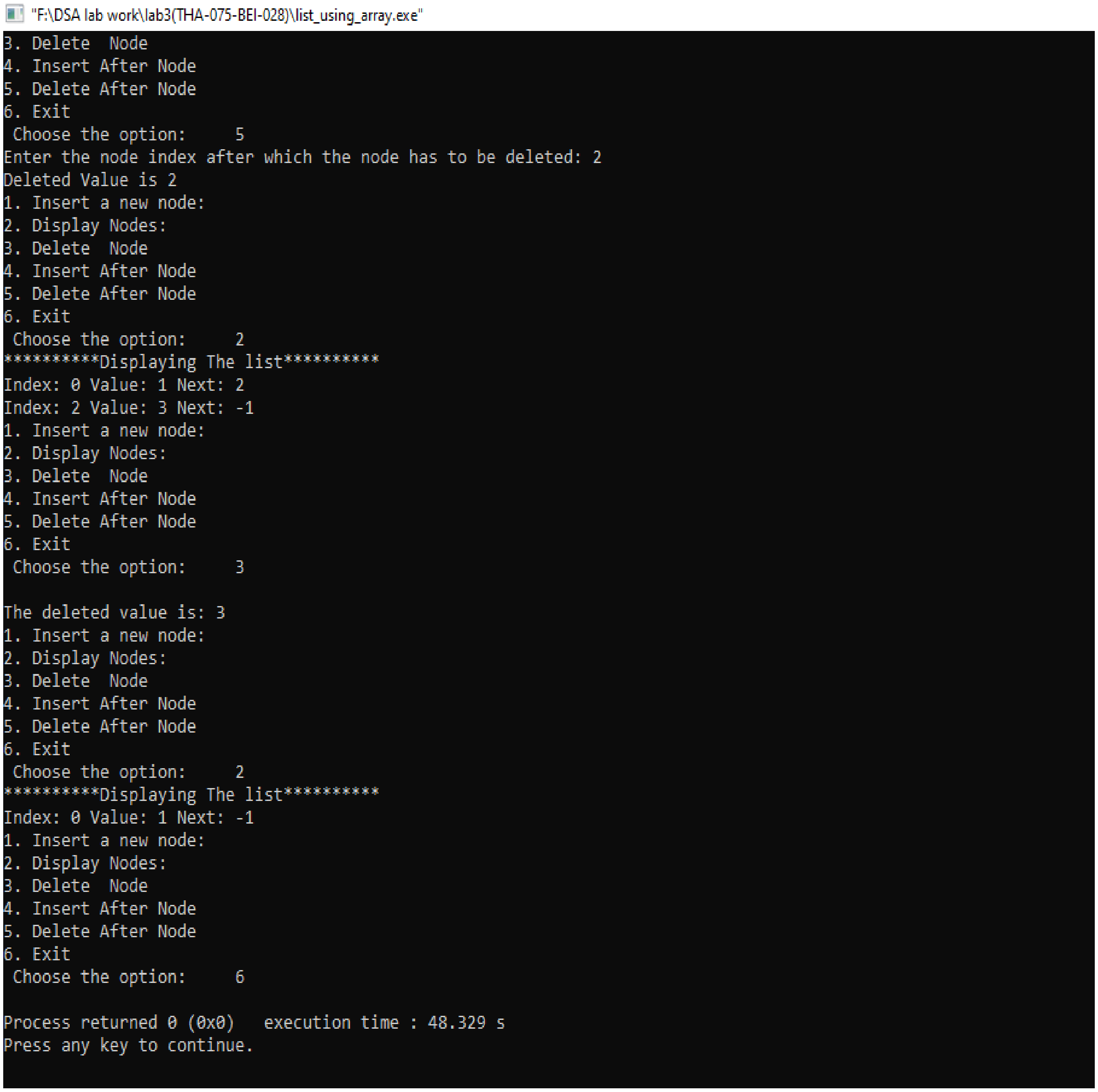
```
cout<<"1. Insert a new node: "<<endl;
cout<<"2. Display Nodes: "<<endl;
cout<<"3. Delete  Node"<<endl;
cout<<"4. Insert After Node"<<endl;
cout<<"5. Delete After Node"<<endl;
cout<<"6. Exit"<<endl;
cout<<" Choose the option: \t";
cin>>ch;
switch(ch)
{
case 1:
        l.insertnode(list1);
        break;
case 2:
        l.displaynode();
        break;
case 3:
        l.deletenode(list1);
        break;
case 4:
        {
        int ptr,val;
        cout<<"Enter the node index after which the new node has to be inserted: ";
        cin>>ptr;
        cout<<"Enter the value to be inserted in the new node: ";
        cin>>val;
        l.insert_after(ptr,val);
        break;
        }
    case 5:
        {
        int ptr;
        cout<<"Enter the node index after which the node has to be deleted: ";
        cin>>ptr;
        l.delete_after(ptr);
        break;
        }
    case 6: break;
    default: cout<<"invalid input"<<endl;
    }
    }while (ch!=6);

}
```

Output :



"F:\DSA lab work\lab3(THA-075-BEI-028)\list_using_array.exe"

```
1. Insert a new node:
2. Display Nodes:
3. Delete  Node
4. Insert After Node
5. Delete After Node
6. Exit
 Choose the option:      1
Enter the number: 1
enter 1 for newnode
1
Enter the Number: 2
enter 1 for newnode
-1
1. Insert a new node:
2. Display Nodes:
3. Delete  Node
4. Insert After Node
5. Delete After Node
6. Exit
 Choose the option:      2
**********Displaying The list**********
Index: 0 Value: 1 Next: 1
Index: 1 Value: 2 Next: -1
1. Insert a new node:
2. Display Nodes:
3. Delete  Node
4. Insert After Node
5. Delete After Node
6. Exit
 Choose the option:      4
Enter the node index after which the new node has to be inserted: 0
Enter the value to be inserted in the new node: 3
Inserted Node After 0 Value: 3 Index: 2
1. Insert a new node:
2. Display Nodes:
3. Delete  Node
4. Insert After Node
5. Delete After Node
6. Exit
 Choose the option:      2
**********Displaying The list**********
Index: 0 Value: 1 Next: 2
Index: 2 Value: 3 Next: 1
Index: 1 Value: 2 Next: -1
1. Insert a new node:
2. Display Nodes:
```



"F:\DSA lab work\lab3(THA-075-BEI-028)\list_using_array.exe"

```
3. Delete  Node
4. Insert After Node
5. Delete After Node
6. Exit
 Choose the option:     5
Enter the node index after which the node has to be deleted: 2
Deleted Value is 2
1. Insert a new node:
2. Display Nodes:
3. Delete  Node
4. Insert After Node
5. Delete After Node
6. Exit
 Choose the option:     2
**********Displaying The list**********
Index: 0 Value: 1 Next: 2
Index: 2 Value: 3 Next: -1
1. Insert a new node:
2. Display Nodes:
3. Delete  Node
4. Insert After Node
5. Delete After Node
6. Exit
 Choose the option:     3

The deleted value is: 3
1. Insert a new node:
2. Display Nodes:
3. Delete  Node
4. Insert After Node
5. Delete After Node
6. Exit
 Choose the option:     2
**********Displaying The list**********
Index: 0 Value: 1 Next: -1
1. Insert a new node:
2. Display Nodes:
3. Delete  Node
4. Insert After Node
5. Delete After Node
6. Exit
 Choose the option:     6

Process returned 0 (0x0)   execution time : 48.329 s
Press any key to continue.
```

Conclusion:

Hence, array implementation of list was programmed with functions to free the node, get the available node, insert, delete, insert after and delete after a node

3.2 WAP for list implementation of queue

Problem analysis:

Queue can be implemented by using static list structure. Initially front and rear nodes can be made null. Operations like enqueue and dequeue can be implemented by getting the available node and making the node free as required.

Algorithm:

Enqueue:

```
Step 1. Input a value val
Step 2. ptr = available node
Step 3. node[ptr].info = val
Step 4 . node[ptr].next = null
Step 5. if rear = null
                Front = ptr
        Else:
                Node[rear].next = ptr
Step 6. rear = ptr
Step 7. Exit
```

Dequeue:

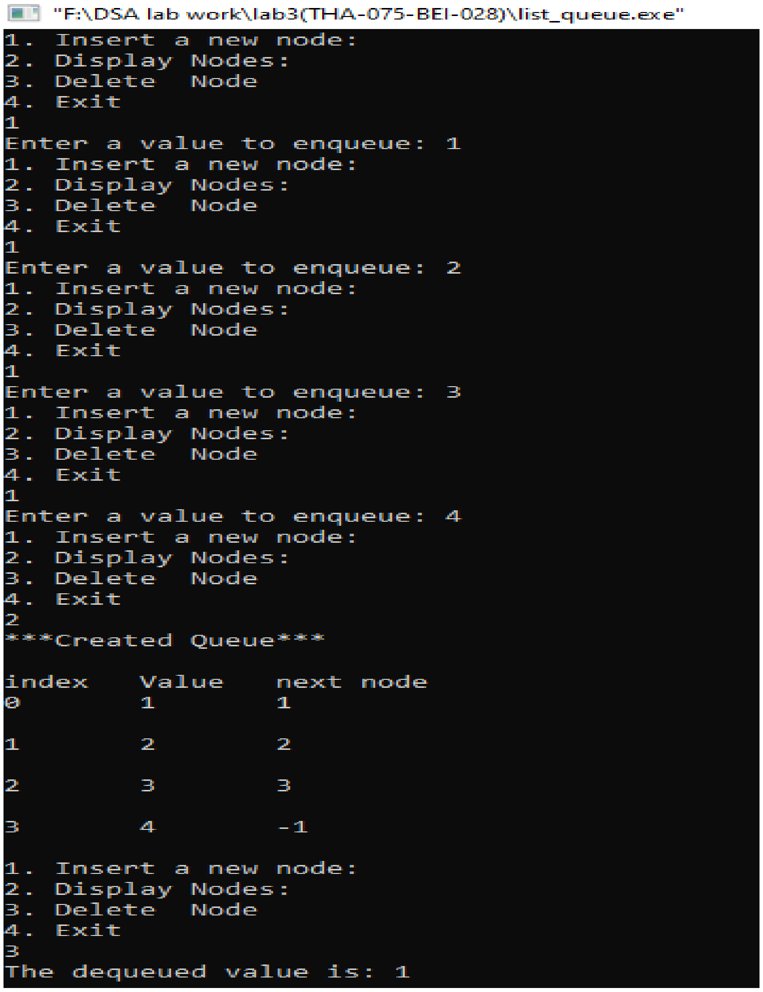
```
Step 1. If front = null or front> rear
        Write “Underflow”
        Goto step
Step 2. delval = node[front].info
Step 3. ptr = front
Step 4. front = node[front].next
Step 5. if front = null
                Rear = null
Step 6. Free node ptr
Step 7. Return delval
Step 8. Exit
```

Source code :

```
//WAP for list implementation of QUEUE
#include<iostream>
#define max 15
using namespace std;
struct nodetype
{
    int info,next;
};
class list
{
    struct nodetype node[max];
    int avail=0;
    int front=-1,rear=-1;
public:
    int intialize_availlist()
    {
        int i;
        for(i=0;i<max-1;i++)
        {
            node[i].next=i+1;
        }
        node[max-1].next=-1;
    }
    int get_node()
    {
        int p;
        if(avail==-1)
        {
            cout<<"Overflow";
        }
        p=avail;
        avail=node[avail].next;
        return p;
    }
    int freenode(int p)
    {
        node[p].next=avail;
        avail=p;
    }
     void enqueue()
    {   int val, ptr;
        if(rear==max-1)
        {
            cout<<"Overflow"<<endl<<endl;
        }
        else
        {
            ptr = get_node();
            cout<<"Enter a value to enqueue: ";
            cin>>val;
            node[ptr].info = val;
            node[ptr].next = -1;
            if(rear == -1)
            {
                front = ptr;
            }
            else
            {
                node[rear].next = ptr;
            }
            rear = ptr;
        }
    }
    void dequeue()
    {
        int val, ptr;
        if(front>rear || front<0)
        {
            cout<<"Underflow"<<endl<<endl;
        }
        else
        {
            val = node[front].info;
            cout<<"The dequeued value is: "<<val<<endl<<endl;
            ptr = front;
            front = node[front].next;
            if(front == -1)
            {
                rear = -1;
            }
            freenode(ptr);
        }
    }
```

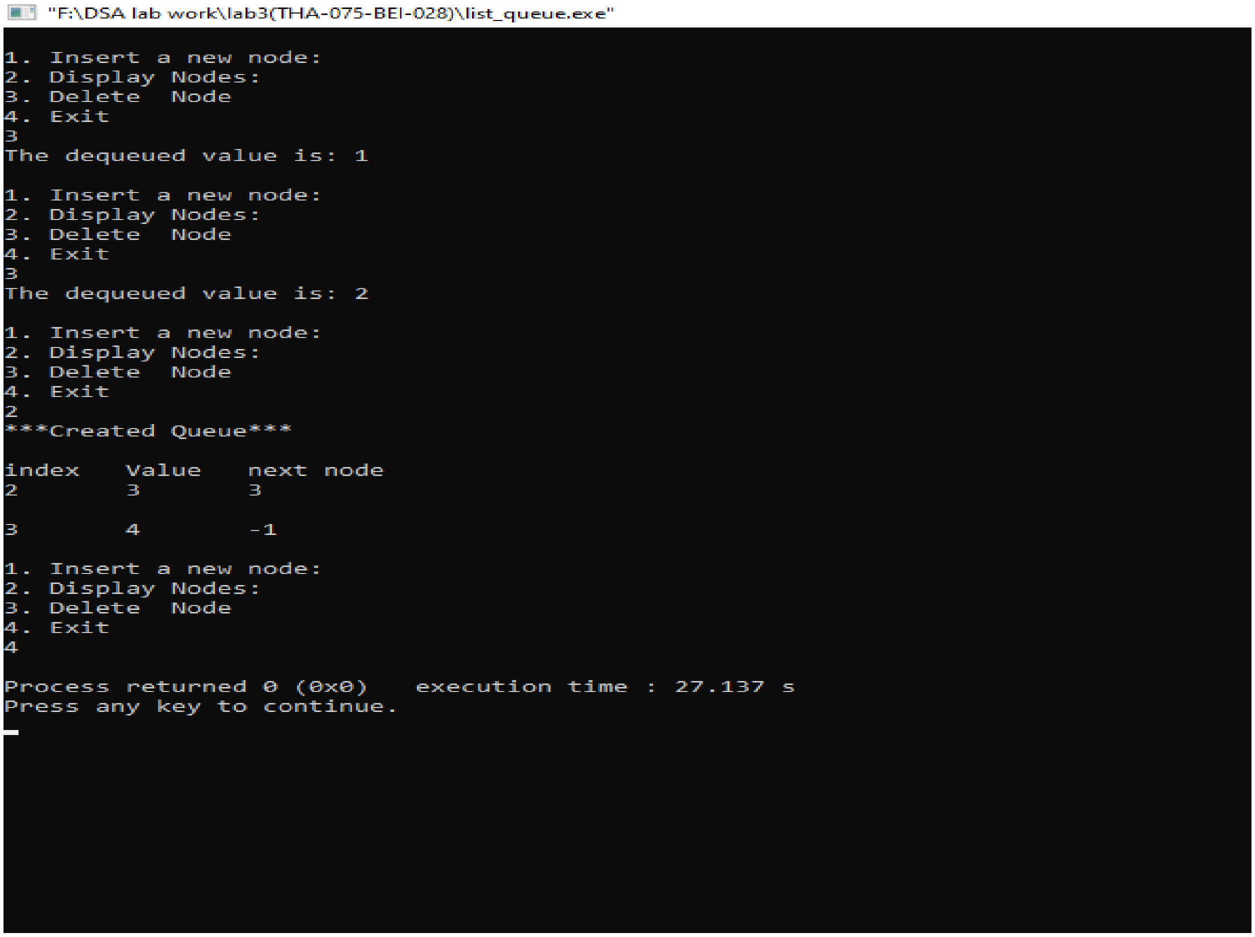
```
    int displaynode()
    {
      cout<<"***Created Queue***"<<endl<<endl;
      cout<<"index\t"<< "Value\t"<<"next node"<<endl;

          for(int i=front ; i <=rear; i++)
          {
             cout<<i<<"\t";
             cout<<node[i].info<<"\t";
             cout<<node[i]. next<<endl<<endl;
          }
    }
};
int main()
{
    list l;
    l.intialize_availlist();
    int ch;
    do
    {
    cout<<"1. Insert a new node: "<<endl;
    cout<<"2. Display Nodes: "<<endl;
    cout<<"3. Delete  Node"<<endl;
    cout<<"4. Exit"<<endl;
    cin>>ch;
    switch(ch)
    {
    case 1:
        l.enqueue();
        break;
    case 2:
        l.displaynode();
        break;
    case 3:
        l.dequeue();
        break;
    case 4: break;
    default: cout<<"invalid input"<<endl;
    }
    }while (ch!=4);
}
```

Output :



"F:\DSA lab work\lab3(THA-075-BEI-028)\list_queue.exe"

```
1. Insert a new node:
2. Display Nodes:
3. Delete  Node
4. Exit
1
Enter a value to enqueue: 1
1. Insert a new node:
2. Display Nodes:
3. Delete  Node
4. Exit
1
Enter a value to enqueue: 2
1. Insert a new node:
2. Display Nodes:
3. Delete  Node
4. Exit
1
Enter a value to enqueue: 3
1. Insert a new node:
2. Display Nodes:
3. Delete  Node
4. Exit
1
Enter a value to enqueue: 4
1. Insert a new node:
2. Display Nodes:
3. Delete  Node
4. Exit
2
***Created Queue***

index    Value    next node
0        1        1

1        2        2

2        3        3

3        4        -1

1. Insert a new node:
2. Display Nodes:
3. Delete  Node
4. Exit
3
The dequeued value is: 1
```



"F:\DSA lab work\lab3(THA-075-BEI-028)\list_queue.exe"

```
1. Insert a new node:
2. Display Nodes:
3. Delete  Node
4. Exit
3
The dequeued value is: 1

1. Insert a new node:
2. Display Nodes:
3. Delete  Node
4. Exit
3
The dequeued value is: 2

1. Insert a new node:
2. Display Nodes:
3. Delete  Node
4. Exit
2
***Created Queue***

index    Value    next node
2        3        3

3        4        -1

1. Insert a new node:
2. Display Nodes:
3. Delete  Node
4. Exit
4

Process returned 0 (0x0)   execution time : 27.137 s
Press any key to continue.
```

Conclusion : In this way, queue was implemented using list. It’s operations enqueue and dequeue were performed by acquiring the available node and making the node free.

LAB 4 Linked List

4.1 WAP to implement singly linked list

Problem analysis:

Here we have to create a node . Each node contains a pointer that points to the next node in the list. The last node's next pointer is empty or Null . Null pointer is represented by -1.

Algorithm:

Create node:

```
Step 1: Input data VAL
Step 2: Create a NEW_NODE
Step 3: SET NEW_NODE =>DATA = VAL
Step 4: IF START = NULL
            SET START=NEW_NODE
            SET END = NEW_NODE
        ELSE
            END=>NEXT=NEW_NODE
            END=NEW_NODE
Step 5 : END=>NEXT = NULL
Step 6: Exit
```

Insert node at Beginning

```
Step 1: Input data VAL
Step 2: Create a NEW_NODE
Step 3: IF NEW_NODE = NULL
         write ERROR IN MEMORY ALLOCATION
        Go to Step 7
 Step 4: SET NEW_NODE => DATA = VAL
Step 5: IF START = NULL
             SET NEW_NODE =>NEXT = NULL
```

```
            ELSE
                    SET NEW_NODE =>NEXT = START
    Step 6: SET START = NEW_NODE
    Step 7: Exit
```

Insert node at End

```
    Step 1: Input data VAL
    Step 2: Create a NEW_NODE
    Step 3: IF NEW_NODE = NULL
            Step 3.1: write ERROR IN MEMORY ALLOCATION
            Step 3.2: Go to Step 9
     Step 4: SET NEW_NODE =>DATA = VAL
    Step 5: SET NEW_NODE =>NEXT = NULL
    Step 6: IF START = NULL
             SET START = NEW_NODE
             Go to step 9
    Step 7: Otherwise, SET PTR = START
            Step 7.1: Repeat Step 7.2 while PTR=> NEXT != NULL
            Step 7.2: SET PTR = PTR=>NEXT
            [END of While Loop]
     Step 8: SET PTR=> NEXT = NEW_NODE
    Step 9: Exit
```

Insert node after a given node

```
    Step 1: Input data VAL
    Step 2: Create a NEW_NODE
    Step 3: IF NEW_NODE = NULL
             Step 3.1: write ERROR IN MEMORY ALLOCATION
            Step 3.2: Go to Step 12
```

```
Step 4: SET NEW_NODE => DATA = VAL
Step 5: SET PTR = START
Step 6: SET PREPTR = PTR
Step 7: Repeat Steps 8 and 9 while PREPTR =>DATA != NUM
Step 8: SET PREPTR = PTR
Step 9: SET PTR = PTR=>NEXT
        [End of while loop]
Step 10: PREPTR=>NEXT = NEW_NODE
Step 11: SET NEW_NODE =>NEXT = PTR
Step 12: Exit
```

Insert node before a given node

```
Step 1: Input data VAL
Step 2: Create a NEW_NODE
Step 3: IF NEW_NODE = NULL
        Step 3.1: write ERROR IN MEMORY ALLOCATION
        Step 3.2: Go to Step 12
Step 4: SET NEW_NODE =>DATA = VAL
Step 5: SET PTR = START
Step 6: SET PREPTR = PTR
Step 7: Repeat Steps 8 and 9 while PTR =>DATA != NUM
Step 8: SET PREPTR = PTR
Step 9: SET PTR = PTR=>NEXT
        [End of while loop]
Step 10: PREPTR=> NEXT = NEW_NODE
Step 11: SET NEW_NODE =>NEXT = PTR
Step 12: Exit
```

Delete first node

```
Step 1: If START = NULL
```

```
        Write ‘underflow’
        Goto Step 5
        [END OF IF]
Step 2 : SET PTR = START
Step 3 : SET START = START=>NEXT
Step 4 : FREE PTR
Step 5 : Exit
```

Delete last node

```
Step 1 : IF START = NULL
        Write ‘Underflow’
        Goto Step 8
        [END OF IF]
Step 2 : SET PTR= START
Step 3 : Repeat Steps 4 and 5 While PTR=>NEXT ! = NULL
Step 4 : SET PREPTR =PTR
Step 5 : SET PTR = PTR=>NEXT
        [END OF LOOP]
Step 6 :  SET PREPTR =>NEXT = NULL
Step 7 : FREE PTR
Step 8 : Exit
```

Delete a given node

```
Step 1 : IF START = NULL
        Write ‘Underflow’
        Goto Step 9
        [END OF IF]
Step 2 : SET PTR= START
Step 3 : Repeat Steps 4 and 5 While PTR=>DATA!=NUM
Step 4 : SET PREPTR =PTR
Step 5 : SET PTR = PTR=>NEXT
        [END OF LOOP]
Step 6 : SET TEMP = PTR
Step 7 : SET PREPTR =>NEXT = PTR => NEXT
Step 8 : FREE TEMP
Step 9 : Exit
```

Source Code:

```
//WAP to implement singly linked list
#include<iostream>
using namespace std;
struct node
{
	int data;
	node *next;
};
class list
{
		private:
		node *Start,*End;
		public:
		list()
		{
			Start=NULL;
			End=NULL;
		}
		void createnode()
		{
```

```
int value;
cout<<"enter -1 to end"<<endl;
cout<<"enter the data"<<"\t";
cin>>value;
do {
node *newnode=new node;
                newnode->data=value;
                if(Start==NULL)
                {
                   Start=newnode;
                   End=newnode;
                }
                else
                      {
                         End->next = newnode;
                         End=newnode;
                      }
End->next=NULL;
cout<<"enter the data"<<"\t";
cin>>value;
  }while(value!=-1);
        }
void display()
        {
              cout<<"--------------------------------------------------\n";
              cout<<"---------------Displaying All nodes---------------";
              cout<<"\n--------------------------------------------------\n";
              struct node *temp;
```

```
          temp=Start;
          if(temp==NULL)
        {
           cout<<"Empty linear linked list"<<endl;
        }
        else
        {
           while(temp!=NULL)
           {
              cout<<"\t"<<temp->data;
              temp=temp->next;
           }
           cout<<endl;
        }
}
void insert_start()
{
        cout<<"--------------------------------------------------\n";
        cout<<"----------------Inserting At Start----------------";
        cout<<"\n--------------------------------------------------\n";
        int value;
        cout<<"enter the data to add at start"<<endl;
        cin>>value;
                        node *newnode=new node;
                        newnode->data=value;
                        newnode->next=Start;
                        Start=newnode;
}
```

```
void insert_end()
{
        cout<<"--------------------------------------------------\n";
        cout<<"----------------Inserting At End--------------------";
        cout<<"\n--------------------------------------------------\n";
        int value;
        cout<<"enter the data "<<endl;
        cin>>value;
        struct node *pre,*cur;
                        node *newnode=new node;
                        cur=Start;
                        while(cur!=NULL)
                        {
                                pre=cur;
                                cur=cur->next;
                        }
                        newnode->data=value;
                        pre->next=newnode;
                        newnode->next=cur;

}
void insert_val_before()
{
       cout<<"--------------------------------------------------\n";
        cout<<"-------------Inserting node before given node data--";
        cout<<"\n--------------------------------------------------\n";
        int val,value;
        cout<<"enter the data to add the node before"<<endl;
```

```
    cin>>val;
    cout<<"enter the value "<<endl;
    cin>>value;
            struct node *pre,*cur;
            node *newnode=new node;
            cur=Start;
            while(cur->data!=val)
            {
                pre=cur;
                cur=cur->next;
            }
            newnode->data=value;
            pre->next=newnode;
            newnode->next=cur;
}
void insert_val_after()
{
    cout<<"--------------------------------------------------\n";
    cout<<"-------------Inserting node after given node data---";
    cout<<"\n--------------------------------------------------\n";
    int num,value;
    cout<<"enter the data to add the node after"<<endl;
    cin>>num;
    cout<<"enter the data "<<endl;
    cin>>value;
            struct node *pre,*cur;
            node *newnode=new node;
            cur=Start;
            while(pre->data!=num)
            {
                pre=cur;
                cur=cur->next;
            }
            newnode->data=value;
            pre->next=newnode;
            newnode->next=cur;
}
void delete_first()
{
    cout<<"--------------------------------------------------\n";
    cout<<"----------------Deleting At Start-----------------";
    cout<<"\n--------------------------------------------------\n";
            struct node *temp;
            temp=Start;
            if(temp==NULL)
    {
      cout<<"empty list "<<endl;
    }
    else {
            Start=Start->next;
            cout<<temp->data << " is deleted "<<endl;
            delete temp; }
        }
        void delete_last()
        {
    cout<<"--------------------------------------------------\n";
```

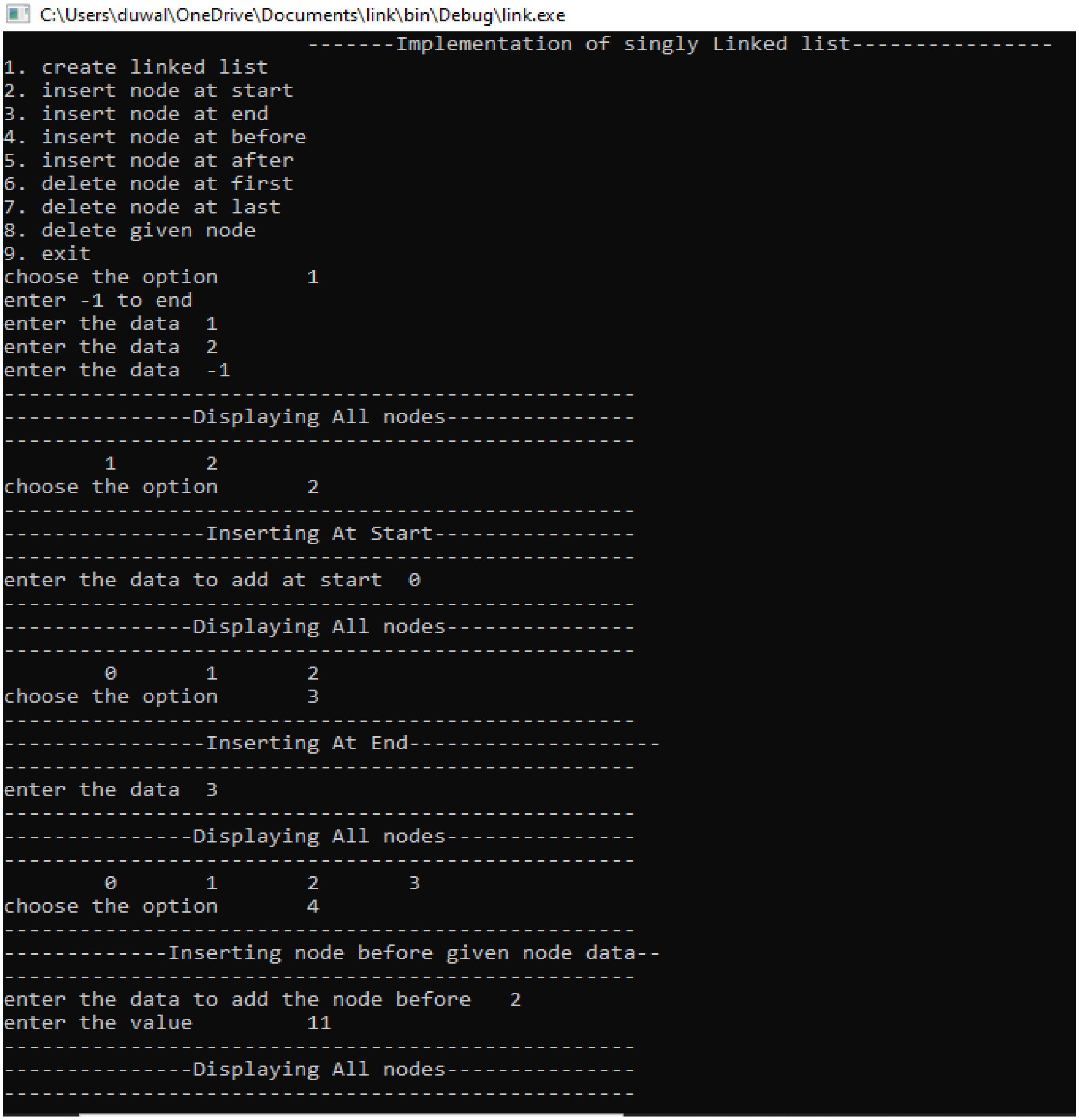
```
            cout<<"-----------------Deleting At End-------------------";
            cout<<"\n----------------------------------------------------\n";
                                    struct node *previous,*current;
                                    current=Start;
                                    if (current == NULL )
            {
               cout<<"empty list "<<endl;
            }
            else {
               while(current->next!=NULL)
               {
                  previous=current;
                  current=current->next;
               }
               previous->next=NULL;
               cout<< current->data <<" is deleted "<<endl;
               delete current;
                 }
}
void delete_num_node()
{
            cout<<"----------------------------------------------------\n";
            cout<<"--------------Deleting given node data----------";
            cout<<"\n----------------------------------------------------\n";
            int num;
            cout<<"enter the node data you want to delete"<<endl;
            cin>>num;
            struct node *previous,*current;
```

```
                    current=Start;
                    if(current==NULL)
                  {
                     cout<<"Empty list  "<<endl;
                  }
                  else
                  {
                     while(current->data!=num)
                               {
                                     previous=current;
                                     current=current->next;
                               }
                               previous->next=current->next;
                               cout<< current->data <<" is deleted "<<endl;
                               delete current;
                  }
                }
};
int main()
{
    list obj;
    int a;
    cout<<"\t\t\t-------Implementation of singly Linked list----------------        "<<endl;
    do{
    cout<<"1. create linked list "<<endl;
    cout<<"2. insert node at start"<<endl;
    cout<<"3. insert node at end"<<endl;
    cout<<"4. insert node at before"<<endl;
```

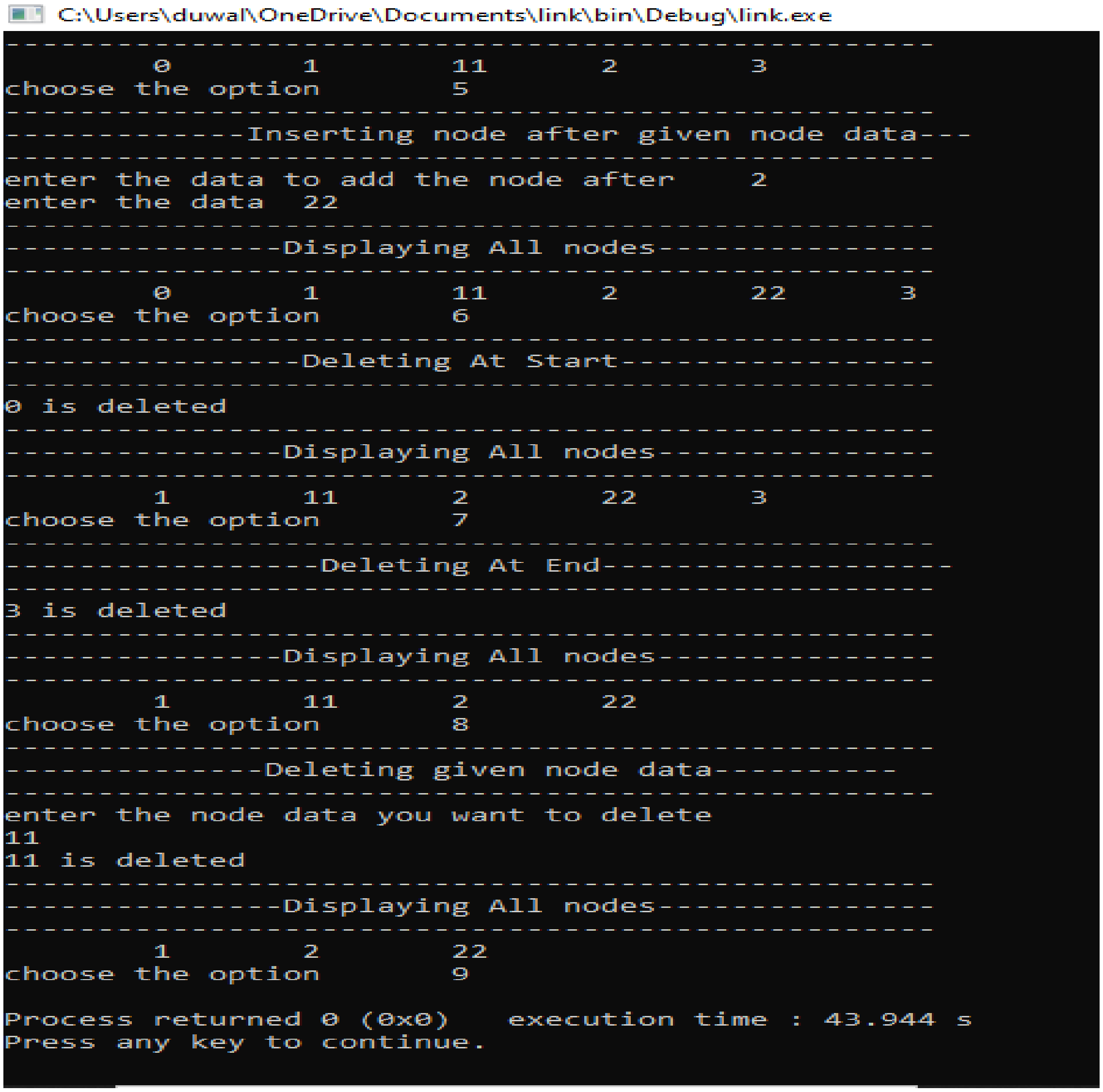
```
cout<<"5. insert node at after"<<endl;
cout<<"6. delete node at first"<<endl;
cout<<"7. delete node at last"<<endl;
cout<<"8. delete given node"<<endl;
    cout<<"9. exit"<<endl;
    cout<<"choose the option"<<endl;
    cin>>a;
    switch(a)
    {
    case 1:
    obj.createnode();
    obj.display();
    break;
case 2:
    obj.insert_start();
    obj.display();
    break;
case 3:
    obj.insert_end();
    obj.display();
    break;
case 4:
    obj.insert_val_before();
    obj.display();
    break;
case 5:
    obj.insert_val_after();
    obj.display();
```

```
          break;
    case 6:
          obj.delete_first();
          obj.display();
          break;
    case 7:
          obj.delete_last();
          obj.display();
          break;
    case 8:
          obj.delete_num_node();
          obj.display();
          break;
    case 9: break;
    default:cout<<"invalid input"<<endl;
    }
    }
    while(a!=9);
}
```

Output :



```
                        -------Implementation of singly Linked list----------------
1. create linked list
2. insert node at start
3. insert node at end
4. insert node at before
5. insert node at after
6. delete node at first
7. delete node at last
8. delete given node
9. exit
choose the option       1
enter -1 to end
enter the data  1
enter the data  2
enter the data  -1
--------------------------------------------------
---------------Displaying All nodes---------------
--------------------------------------------------
        1       2
choose the option       2
--------------------------------------------------
----------------Inserting At Start----------------
--------------------------------------------------
enter the data to add at start  0
--------------------------------------------------
---------------Displaying All nodes---------------
--------------------------------------------------
        0       1       2
choose the option       3
--------------------------------------------------
----------------Inserting At End--------------------
--------------------------------------------------
enter the data  3
--------------------------------------------------
---------------Displaying All nodes---------------
--------------------------------------------------
        0       1       2       3
choose the option       4
--------------------------------------------------
-------------Inserting node before given node data--
--------------------------------------------------
enter the data to add the node before   2
enter the value         11
--------------------------------------------------
---------------Displaying All nodes---------------
--------------------------------------------------
```



C:\Users\duwal\OneDrive\Documents\link\bin\Debug\link.exe

```
--------------------------------------------------
        0       1       11      2       3
choose the option       5
--------------------------------------------------
-------------Inserting node after given node data---
--------------------------------------------------
enter the data to add the node after    2
enter the data  22
--------------------------------------------------
---------------Displaying All nodes---------------
--------------------------------------------------
        0       1       11      2       22      3
choose the option       6
--------------------------------------------------
----------------Deleting At Start-----------------
--------------------------------------------------
0 is deleted
--------------------------------------------------
---------------Displaying All nodes---------------
--------------------------------------------------
        1       11      2       22      3
choose the option       7
--------------------------------------------------
-----------------Deleting At End-------------------
--------------------------------------------------
3 is deleted
--------------------------------------------------
---------------Displaying All nodes---------------
--------------------------------------------------
        1       11      2       22
choose the option       8
--------------------------------------------------
--------------Deleting given node data----------
--------------------------------------------------
enter the node data you want to delete
11
11 is deleted
--------------------------------------------------
---------------Displaying All nodes---------------
--------------------------------------------------
        1       2       22
choose the option       9

Process returned 0 (0x0)   execution time : 43.944 s
Press any key to continue.
```

Conclusion :

In this way single linked list was implemented. By using above code linked list can be create , delete ,insert as per user required.

4.2 WAP to implement circular linked list

Problem analysis :

Here we have to create a node . Each node contains a pointer that points to the next node in the list. The last node's next pointer is pointed to the first node .

Algorithm :

```
            Create node:
            Step 1: Input data VAL
            Step 2: Create a NEW_NODE
            Step 3: SET NEW_NODE =>DATA = VAL
            Step 4: IF START = NULL
                        SET START=NEW_NODE
                        SET END = NEW_NODE
                ELSE
                        END=>NEXT=NEW_NODE
                        END=NEW_NODE
            Step 5 : END=>NEXT = START
            Step 6 : Exit
Insert node at Beginning
            Step 1: Input data VAL
            Step 2: Create a NEW_NODE
            Step 3: IF NEW_NODE = NULL
                 write ERROR IN MEMORY ALLOCATION
                Go to Step 7
             Step 4: SET NEW_NODE => DATA = VAL
            Step 5 : SET PTR = START
            Step 6 : Repeat Step 7
                While PTR=>NEXT!=START
            Step 7 : SET PTR=PTR=>NEXT
```

```
        [END OF While LOOP]
    Step 8: SET NEW_NODE =>NEXT = START
    Step 9 : PTR=>NEXT = NEW_NODE
    Step 10 : SET START = NEW_NODE
    Step 11 : Exit
```

Insert node at End

```
    Step 1: Input data VAL
    Step 2: Create a NEW_NODE
    Step 3: IF NEW_NODE = NULL
        Step 3.1: write ERROR IN MEMORY ALLOCATION
        Step 3.2: Go to Step 9
     Step 4: SET NEW_NODE =>DATA = VAL
    Step 5: SET NEW_NODE =>NEXT = START
    Step 6: SET PTR=START
    Step 7: Repeat Step 8
        While PTR=>NEXT!=START
    Step 8 : SET PTR = PTR=>NEXT
        [END of While Loop]
     Step 9: SET PTR=>NEXT = NEW_NODE
    Step 10: Exit
```

Insert node after a given node

```
    Step 1: Input data VAL
    Step 2: Create a NEW_NODE
    Step 3: IF NEW_NODE = NULL
         Step 3.1: write ERROR IN MEMORY ALLOCATION
        Step 3.2: Go to Step 12
     Step 4: SET NEW_NODE => DATA = VAL
```

```
Step 5: SET PTR = START
Step 6: SET PREPTR = PTR
 Step 7: Repeat Steps 8 and 9 while PREPTR =>DATA != NUM
 Step 8: SET PREPTR = PTR
Step 9: SET PTR = PTR=>NEXT
            [End of while loop]
 Step 10: PREPTR=> NEXT = NEW_NODE
Step 11: SET NEW_NODE =>NEXT = PTR
Step 12: Exit
```

Insert node before a given node

```
Step 1: Input data VAL
Step 2: Create a NEW_NODE
Step 3: IF NEW_NODE = NULL
             Step 3.1: write ERROR IN MEMORY ALLOCATION
            Step 3.2: Go to Step 12
 Step 4: SET NEW_NODE =>DATA = VAL
Step 5: SET PTR = START
Step 6: SET PREPTR = PTR
 Step 7: Repeat Steps 8 and 9 while PTR =>DATA != NUM
 Step 8: SET PREPTR = PTR
Step 9: SET PTR = PTR=>NEXT
            [End of while loop]
 Step 10: PREPTR=>NEXT = NEW_NODE
Step 11: SET NEW_NODE =>NEXT = PTR
Step 12: Exit
```

Delete first node

```
Step 1: If START = NULL
```

```
                Write ‘underflow’
                Goto Step 5
                [END OF IF]
        Step 2 : SET PTR = START
        Step 3 : Repeat Step 4 While PTR=>NEXT != START
        Step 4 : SET PTR=PTR=>NEXT
                [END OF While LOOP]
        Step 5 : SET PTR=>NEXT=START=>NEXT
        Step 6 : FREE START
        Step 7: SET START=PTR=>NEXT
        Step 8 :Exit
```

Delete last node

```
        Step 1 : IF START = NULL
                Write ‘Underflow’
                Goto Step 8
                [END OF IF]
        Step 2 : SET PTR= START
        Step 3 : Repeat Steps 4 and 5 While PTR=>NEXT ! = NULL
        Step 4 : SET PREPTR =PTR
        Step 5 : SET PTR = PTR=>NEXT
                [END OF LOOP]
        Step 6 :  SET PREPTR =>NEXT = START
        Step 7 : FREE PTR
        Step 8 : Exit
```

Delete a given node

```
        Step 1 : IF START = NULL
                Write ‘Underflow’
                Goto Step 9
```

```
            [END OF IF]
        Step 2 : SET PTR= START
        Step 3 : Repeat Steps 4 and 5 While PTR=>DATA!=NUM
        Step 4 : SET PREPTR =PTR
        Step 5 : SET PTR = PTR=>NEXT
            [END OF LOOP]
        Step 6 : SET TEMP = PTR
        Step 7 : SET PREPTR =>NEXT = PTR => NEXT
        Step 8 : FREE TEMP
        Step 9 : Exit
```

Source Code :

```
//WAP to implement circular linked list
#include<iostream>
using namespace std;
struct node
{
   int data;
   node* next;
};

class circularlinkedlist
{
   node *head;
public:
   circularlinkedlist()
   {
      head = NULL;
   }
```

```
void create_circularlinkedlist()
{
    int val;
    cout<<"enter -1 to end"<<endl;
    cout<<"Enter a value: ";
    cin>>val;
    do
    {
        node *newNode = new node;
        newNode->data = val;
        if(head == NULL)
        {
           head= newNode;
           newNode->next = head;
        }
        else
        {
           node *ptr = head;
           while(ptr->next != head)
           {
               ptr = ptr->next;
           }
           ptr->next = newNode;
           newNode->next = head;
        }
        cout<<"Enter a value: ";
        cin>>val;
    }while(val != -1);
}
void insert_end_circularlinkedlist()
{
    int val;
    cout<<"\nenter the number to insert at end: ";
    cin>>val;
    node *ptr = head;
    node *newNode = new node;
    newNode->data = val;
    newNode->next = head;
    if(head == NULL)
    {
        head= newNode;
    }
    else
    {
        while(ptr->next != head)
        {
            ptr = ptr->next;
        }
        ptr->next = newNode;
        newNode->next = head;
    }
}

void insert_beg_circularlinkedlist(int val)
{
    node *newNode = new node;
    newNode->data = val;
    newNode->next = head;
    node *ptr = head;
    while(ptr->next != head)
    {
        ptr = ptr->next;
    }
    ptr->next = newNode;
    head = newNode;
}

void insert_before_circularlinkedlist()
{
    int n,val;
    cout<<"--------------------------------------------------\n";
    cout<<"-------------Inserting node before given node data--";
    cout<<"\n--------------------------------------------------\n";
    cout<<"enter the data to add the node before"<<endl;
    cin>>n;
    cout<<"enter the value "<<endl;
    cin>>val;
    node *newNode = new node;
    newNode->data = val;
    if(head->data == n)
    {
        insert_beg_circularlinkedlist(n);
    }
    else
    {
        node *ptr = head;
        node *preptr;

        while(ptr->data != n)
        {
            preptr = ptr;
            ptr = ptr->next;
        }

        preptr->next = newNode;
        newNode->next = ptr;
    }
}
void insert_after_circularlinkedlist()
{
    int n,val;
    cout<<"--------------------------------------------------\n";
    cout<<"-------------Inserting node after given node data--";
    cout<<"\n--------------------------------------------------\n";
    cout<<"enter the data to add the node after"<<endl;
    cin>>n;
    cout<<"enter the value "<<endl;
    cin>>val;
    node *newNode = new node;
    newNode->data = val;
    node *ptr = head;
    while (ptr->data != n)
    {
        ptr = ptr->next;
    }
    if(ptr->next == head)
    {
      ptr->next = newNode;
      newNode->next = head;
    }
    else
    {
        newNode->next=ptr->next;
        ptr->next = newNode;
    }
}

void delete_beg_circularlinkedlist()
{
    if(head->next == head)
    {
        head = NULL;
    }
    else
    {
        node *ptr = head;
        node *tmp = head;
        while(ptr->next != head)
        {
            ptr = ptr->next;
        }
        ptr->next = head->next;
        head = head->next;
        cout<<tmp->data<<" is deleted"<<endl;
        delete tmp;
    }
}

void delete_end_circularlinkedlist()
{
    node *ptr =head;
    node *preptr = ptr;
    while(ptr->next != head)
    {
        preptr = ptr;
        ptr = ptr->next;
    }
    preptr->next = head;
    cout<<ptr->data<<" is deleted"<<endl;
    delete ptr;
}

void delete_node_circularlinkedlist()
{
    cout<<"--------------------------------------------------\n";
    cout<<"--------------Deleting given node data----------";
    cout<<"\n--------------------------------------------------\n";
    int n;
    cout<<"enter the node data you want to delete"<<endl;
    cin>>n;
    node *ptr = head;
    if(ptr->data == n)
    {
        delete_beg_circularlinkedlist();
    }
    else
    {
        node*preptr = ptr;
        while(ptr->data != n)
        {
            preptr = ptr;
            ptr = ptr->next;
        }
        preptr->next = ptr->next;
        cout<<ptr->data<<" is deleted"<<endl;
        delete ptr;
    }
}
void delete_after_circularlinkedlist()
{
    cout<<"--------------------------------------------------\n";
    cout<<"--------------Deleting after node data----------";
    cout<<"\n--------------------------------------------------\n";
    int n;
    cout<<"enter the node data"<<endl;
    cin>>n;
    node *ptr= head;
    while(ptr->data != n)
    {
        ptr = ptr->next;
    }
    if(ptr->next == head)
    {
        delete_beg_circularlinkedlist();
    }
    else
    {
        node *tmp = ptr->next;
        ptr->next = tmp->next;
        cout<<tmp->data<<" is deleted"<<endl;
        delete tmp;
    }
}

void delete_circularlinkedlist()
{
    cout<<"--------------------------------------------------\n";
    cout<<"--------Deleting whole circular linked list---------\n";
    cout<<"\n--------------------------------------------------\n";
    while(head != NULL)
    {
        delete_beg_circularlinkedlist();
        }
    }
    void display_circularlinkedlist()
    {
        cout<<"--------------------------------------------------\n";
        cout<<"---------------Displaying All nodes---------------";
        cout<<"\n--------------------------------------------------\n";
        node *ptr = head;
        if(head == NULL)
        {
            cout<<"\nThe list is empty!!"<<endl;
        }
        else
        {
            while(ptr->next != head)
            {
                cout<<" "<<ptr->data<<" ";
                ptr = ptr->next;
            }
            cout<<" "<<ptr->data<<" ";
            cout<<endl;
        }
    }

};

int main()
{
```

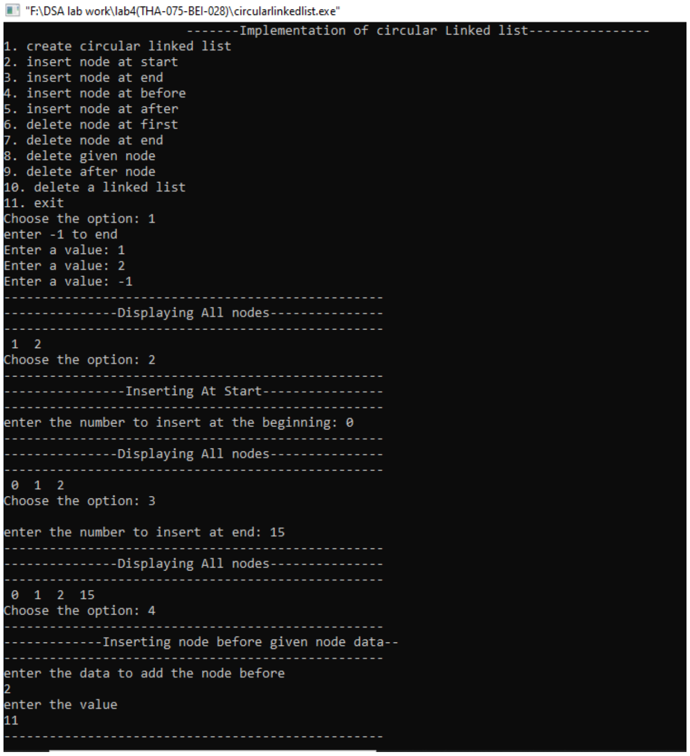
```
circularlinkedlist c;
int ch;
cout<<"\t\t\t-------Implementation of circular Linked list----------------        "<<endl;
do
{
   cout<<"1. create circular linked list "<<endl;
   cout<<"2. insert node at start"<<endl;
   cout<<"3. insert node at end"<<endl;
   cout<<"4. insert node at before"<<endl;
   cout<<"5. insert node at after"<<endl;
   cout<<"6. delete node at first"<<endl;
   cout<<"7. delete node at end"<<endl;
   cout<<"8. delete given node"<<endl;
   cout<<"9. delete after node"<<endl;
   cout<<"10. delete a linked list"<<endl;
   cout<<"11. exit"<<endl;
   cout<<"Choose the option: ";
   cin>>ch;
   switch (ch)
   {
   case 1:
   {
      c.create_circularlinkedlist();
      c.display_circularlinkedlist();
      break;
   }
   case 2:
   {
    int val;
    cout<<"--------------------------------------------------\n";
    cout<<"----------------Inserting At Start----------------";
    cout<<"\n--------------------------------------------------\n";
    cout<<"enter the number to insert at the beginning: ";
    cin>>val;
    c.insert_beg_circularlinkedlist(val);
    c.display_circularlinkedlist();
    break;
}
case 3:
{
    c.insert_end_circularlinkedlist();
    c.display_circularlinkedlist();
    break;
}
case 4:
{

    c.insert_before_circularlinkedlist();
    c.display_circularlinkedlist();
    break;
}

case 5:
{
    c.insert_after_circularlinkedlist();
    c.display_circularlinkedlist();
```

```
   break;
}

case 6:
{
   c.delete_beg_circularlinkedlist();
   c.display_circularlinkedlist();
   break;
}

case 7:
{
   c.delete_end_circularlinkedlist();
   c.display_circularlinkedlist();
   break;
}

case 8:
{
   c.delete_node_circularlinkedlist();
   c.display_circularlinkedlist();
   break;
}

case 9:
{
   c.delete_after_circularlinkedlist();
   c.display_circularlinkedlist();
```

```
            break;
        }

        case 10:
        {
                c.delete_circularlinkedlist();
                c.display_circularlinkedlist();
                break;
        }

        case 11:
        {
            break;
        }

        default :
        {
            cout<<"Invalid input";
            break;
        }
        }
    }while (ch!=11);
    return 0;
}
```

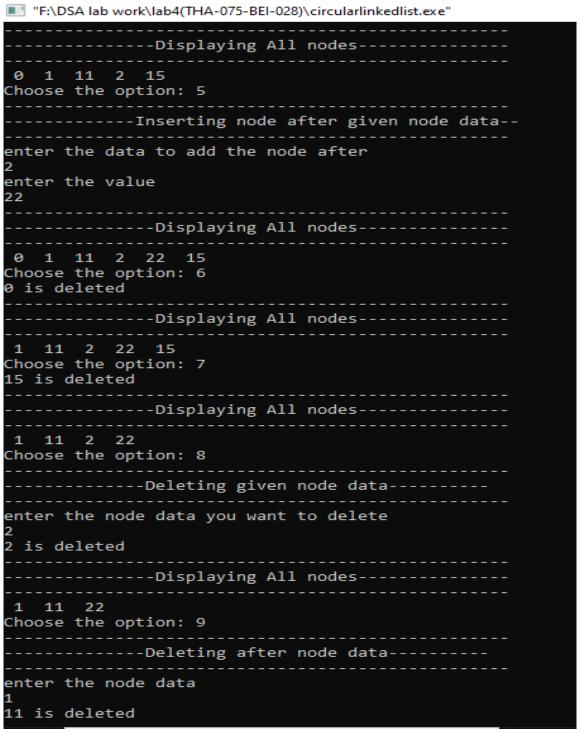
Output:

1.



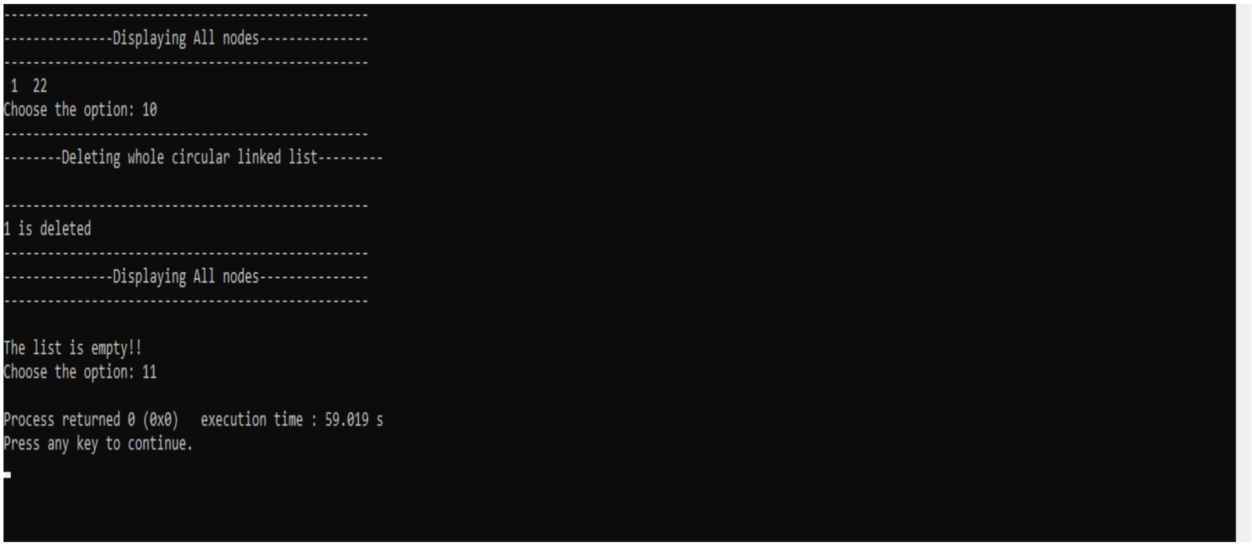
```
"F:\DSA lab work\lab4(THA-075-BEI-028)\circularlinkedlist.exe"
                        -------Implementation of circular Linked list---------------
1. create circular linked list
2. insert node at start
3. insert node at end
4. insert node at before
5. insert node at after
6. delete node at first
7. delete node at end
8. delete given node
9. delete after node
10. delete a linked list
11. exit
Choose the option: 1
enter -1 to end
Enter a value: 1
Enter a value: 2
Enter a value: -1
--------------------------------------------------
--------------Displaying All nodes---------------
--------------------------------------------------
 1  2
Choose the option: 2
--------------------------------------------------
---------------Inserting At Start----------------
--------------------------------------------------
enter the number to insert at the beginning: 0
--------------------------------------------------
--------------Displaying All nodes---------------
--------------------------------------------------
 0  1  2
Choose the option: 3

enter the number to insert at end: 15
--------------------------------------------------
--------------Displaying All nodes---------------
--------------------------------------------------
 0  1  2  15
Choose the option: 4
--------------------------------------------------
------------Inserting node before given node data--
--------------------------------------------------
enter the data to add the node before
2
enter the value
11
--------------------------------------------------
```

2.



```
"F:\DSA lab work\lab4(THA-075-BEI-028)\circularlinkedlist.exe"
--------------------------------------------------
---------------Displaying All nodes---------------
--------------------------------------------------
 0  1  11  2  15
Choose the option: 5
--------------------------------------------------
-------------Inserting node after given node data--
--------------------------------------------------
enter the data to add the node after
2
enter the value
22
--------------------------------------------------
---------------Displaying All nodes---------------
--------------------------------------------------
 0  1  11  2  22  15
Choose the option: 6
0 is deleted
--------------------------------------------------
---------------Displaying All nodes---------------
--------------------------------------------------
 1  11  2  22  15
Choose the option: 7
15 is deleted
--------------------------------------------------
---------------Displaying All nodes---------------
--------------------------------------------------
 1  11  2  22
Choose the option: 8
--------------------------------------------------
--------------Deleting given node data----------
--------------------------------------------------
enter the node data you want to delete
2
2 is deleted
--------------------------------------------------
---------------Displaying All nodes---------------
--------------------------------------------------
 1  11  22
Choose the option: 9
--------------------------------------------------
--------------Deleting after node data----------
--------------------------------------------------
enter the node data
1
11 is deleted
```

3.



```
------------------------------------------------
--------------Displaying All nodes--------------
------------------------------------------------
 1  22
Choose the option: 10
------------------------------------------------
--------Deleting whole circular linked list--------

------------------------------------------------
1 is deleted
------------------------------------------------
--------------Displaying All nodes--------------
------------------------------------------------


The list is empty!!
Choose the option: 11

Process returned 0 (0x0)   execution time : 59.019 s
Press any key to continue.
```

Conclusion :

In this way Circular linked list was implemented. By using above code circular linked list can be create , delete ,insert as per user required.

4.3 WAP to implement doubly linked list

Problem analysis:
Here we have to create a node . Each node contains a pointer that points to the next node in the list. The last node's next pointer is pointed to the first node. Since it is doubly linked list or a two-way linked list which is a more complex type of linked list which contains a pointer to the next as well as the previous node in the sequence. The problem here is to linked previous and current node. i.e we have to create such type of linked list where not only forward but also backward traversing possible.

Algorithm:

Inserting at beginning:

```
Step 1: Input data VAL
Step 2: Create a NEW_NODE
Step 3: IF NEW_NODE = NULL
    Step 3.1: write ERROR IN MEMORY ALLOCATION
    Step 3.2: Go to Step 9
Step 4: SET NEW_NODE => DATA = VAL
Step 5: SET NEW_NODE =>PREV = NULL
Step 6: SET NEW_NODE => NEXT = START
Step 7: SET START=>PREV = NEW_NODE
Step 8: SET START = NEW_NODE
Step 9: EXIT
```

Inserting at End

```
Step 1: Input data VAL
Step 2: Create a NEW_NODE
Step 3: IF NEW_NODE = NULL
    Step 3.1: write ERROR IN MEMORY ALLOCATION
    Step 3.2: Go to Step 11
Step 4: SET NEW_NODE => DATA = VAL
Step 5: SET NEW_NODE =>NEXT = NULL
Step 6: SET PTR = START
Step 7: Repeat Step 8 While PTR=> NEXT != NULL
Step 8: SET PTR = PTR=>NEXT
Step 9: SET PTR => NEXT = NEW_NODE
Step 10 : Set NEW_NODE=>PREV = PTR
Step 11: EXIT
```

Inserting node after a given node

```
Step 1: Input data VAL
Step 2: Create a NEW_NODE
Step 3: IF NEW_NODE = NULL
        Step 3.1: write ERROR IN MEMORY ALLOCATION
        Step 3.2: Go to Step 12
Step 4: SET NEW_NODE => DATA = VAL
Step 5: SET PTR = START
Step 6: Repeat Step 7 While PTR=>DATA != NUM
Step 7: SET PTR = PTR=>NEXT
        [End of while Loop]
Step 8: SET NEW_NODE =.>NEXT = PTR ◊ NEXT
Step 9: SET NEW_NODE=>PREV = PTR
Step 10 : SET PTR=>NEXT=>PREV = NEW_NODE
Step 11: SET PTR=>NEXT = NEW_NODE
Step 12: EXIT
```

Inserting node before a given node

```
Step 1: Input data VAL
Step 2: Create a NEW_NODE
Step 3: IF NEW_NODE = NULL
        Step 3.1: write ERROR IN MEMORY ALLOCATION
        Step 3.2: Go to Step 12
Step 4: SET NEW_NODE =>DATA = VAL
Step 5: SET PTR = START
Step 6: Repeat Step 7 While PTR=> DATA != NUM
Step 7: SET PTR = PTR=>NEXT [End of while Loop]
Step 8: SET NEW_NODE =>NEXT = PTR
Step 9: SET NEW_NODE=>PREV = PTR◊PREV
Step 10 : SET PTR=>PREV=>NEXT = NEW_NODE
Step 11: SET PTR=>PREV = NEW_NODE
 Step 12: EXIT
```

Deleting the first node

```
Step 1 : IF START = NULL
            Write “underflow”
            Goto step 6
            [END OF IF]
Step 2 : SET PTR = START
Step 3 : SET START=START=>NEXT
Step 4 : SET START=>PREV=NULL
Step 5 : FREE PTR
Step 6 : EXIT
```

Deleting the last node

```
Step 1 : IF START = NULL
            Write “underflow”
            Goto step 7
            [END OF IF]
Step 2 : SET PTR = START
Step 3 : Repeat Step 4 While PTR=>NEXT!=NULL
Step 4 : SET PTR = PTR=>NEXT
            [ END OF LOOP ]
```

```
Step 5 : SET PTR=>PREV=>NEXT=NULL
Step 6 : FREE PTR
Step 7 : EXIT
```

Deleting node after a given node

```
Step 1 : IF START=NULL
            Write “UNDERFLOW”
            Goto Step 9
            [ END OF IF ]
Step 2 : SET PTR= START
Step 3 : Repeat Step 4 While PTR=>DATA!=NUM
Step 4 : SET PTR=PTR=>NEXT
            [END OF LOOP]
Step 5 : SET TEMP = PTR=>NEXT
Step 6 : SET PTR=>NEXT = TEMP=>NEXT
Step 7 : SET TEMP=>NEXT=>PREV=PTR
Step 8 : FREE TEMP
Step 9: EXIT
```

Deleting node before a given node

```
Step 1 : IF START=NULL
            Write “UNDERFLOW”
            Goto Step 9
            [ END OF IF ]
Step 2 : SET PTR= START
Step 3 : Repeat Step 4 While PTR=>DATA!=NUM
Step 4 : SET PTR=PTR=>NEXT
            [END OF LOOP]
Step 5 : SET TEMP = PTR=>PREV
Step 6 : SET TEMP=>PREV=>NEXT = PTR
```

```
    Step 7 : SET PTR=>PREV = TEMP =>PREV
    Step 8 : FREE TEMP
    Step 9: EXIT
```

Deleting a given node

```
    Step 1 : IF START=NULL
        Write “UNDERFLOW”
        Goto Step 8
        [ END OF IF ]
    Step 2 : SET PTR= START
    Step 3 : Repeat Step 4 While PTR=>DATA!=NUM
    Step 4 : SET PTR=PTR=>NEXT
        [END OF LOOP]
    Step 5 : SET PTR=>PREV=>NEXT=PTR=>NEXT
    Step 6 : SET PTR=>NEXT=>PREV=PTR=>PREV
    Step 7 : FREE PTR
    Step 8: EXIT
```

Source code:

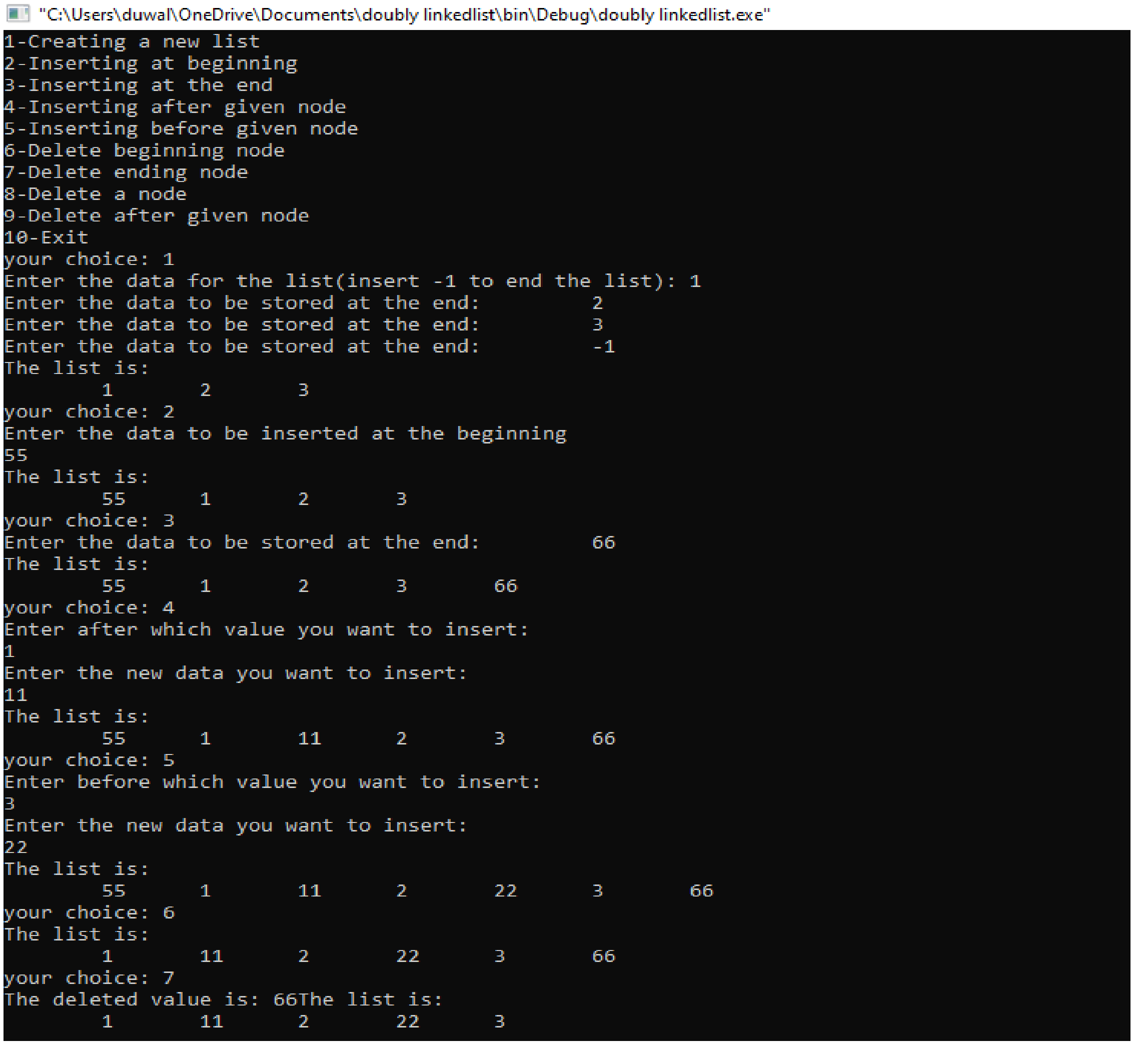
```
#include<iostream>
using namespace std;
struct node
{
   int data;
   struct node * next;
   struct node * prev;
};
struct node * start;
struct node * newnode,* temp,* ptr;
void insert_end();
void creation()
{
        newnode = new node;
        cout<<"Enter the data for the list(insert -1 to end the list): ";
        cin>>newnode->data;
        newnode->prev=NULL;
        newnode->next=NULL;
        if (start==NULL)
        {
            start=newnode;
            temp=newnode;
        }
        else
        {
            temp->next=newnode;
            temp=newnode;
        }
    do{
        insert_end();
    }while (newnode->data!=-1);
}
void insert_end()
{
    newnode=new node;
    cout<<"Enter the data to be stored at the end: \t";
    cin>>newnode->data;
    if (newnode->data!=-1)
    {
        newnode->next=NULL;
        newnode->prev=NULL;
        ptr=start;
        while(ptr->next!=NULL)
        {
            ptr=ptr->next;
        }
        ptr->next=newnode;
        newnode->prev=ptr;
    }
}
void insert_begin()
{
    newnode = new node;
    cout<<"Enter the data to be inserted at the beginning"<<endl;
    cin>>newnode->data;
    newnode->prev=NULL;
    newnode->next=start;
    start->prev=newnode;
    start=newnode;
}
void insert_afternode()
{
    int val;
    newnode = new node;
    cout<<"Enter after which value you want to insert: "<<endl;
    cin>>val;
    cout<<"Enter the new data you want to insert: "<<endl;
    cin>>newnode->data;
    ptr=start;
    while(ptr->data!=val)
    {
        ptr=ptr->next;
        if(ptr==NULL)
        {
            cout<<"error data not found";
        }
    }
    newnode->next=ptr->next;
    newnode->prev=ptr;
    ptr->next->prev=newnode;
    ptr->next=newnode;
}
void insert_beforenode()
{
    int val;
    newnode = new node;
    cout<<"Enter before which value you want to insert: "<<endl;
    cin>>val;
    cout<<"Enter the new data you want to insert: "<<endl;
    cin>>newnode->data;
    ptr=start;
    while(ptr->data!=val)
    {
        ptr=ptr->next;
        if(ptr==NULL)
        {
            cout<<"error data not found";
        }
    }
    newnode->next=ptr;
    newnode->prev=ptr->prev;
    ptr->prev->next=newnode;
    ptr->prev=newnode;
}
void del_end()
{
    ptr=start;
    while(ptr->next!=NULL)
    {
        ptr=ptr->next;
    }
    cout<<"The deleted value is: "<<ptr->data;
    ptr->prev->next=NULL;
    delete ptr;
}
void del_begin()
{
    ptr=start->next;
    delete start;
    start=ptr;
    ptr->prev=NULL;
}
void del_node()
```

```
{
    int val;
    cout<<"Enter the value of node which you want to delete: "<<endl;
    cin>>val;
    ptr=start;
    while(ptr->data!=val)
    {
        ptr=ptr->next;
    }
    ptr->prev->next=ptr->next;
    ptr->next->prev=ptr->prev;
    delete ptr;
}
void del_after ()
{
    int val;
    cout<<"Enter the value of node after which you want to delete: "<<endl;
    cin>>val;
    ptr=start;
    while(ptr->data!=val)
    {
        ptr=ptr->next;
    }
    temp=ptr->next;
    ptr->next=temp->next;
    temp->next->prev=ptr;
    delete temp;
}
```

```
void display_list()
{
    ptr=start;
    cout<<"The list is: "<<endl;
    cout<<"\t"<<ptr->data;
    while(ptr->next!=NULL)
    {
        ptr=ptr->next;
        if (ptr->data==-1)
            break;
        cout<<"\t"<<ptr->data;
    }
    cout<<endl;
}
int main()
{
    start=NULL;
    int choice;
    cout<<"1-Creating a new list "<<endl;
    cout<<"2-Inserting at beginning "<<endl;
    cout<<"3-Inserting at the end "<<endl;
    cout<<"4-Inserting after given node "<<endl;
    cout<<"5-Inserting before given node"<<endl;
    cout<<"6-Delete beginning node "<<endl;
    cout<<"7-Delete ending node "<<endl;
    cout<<"8-Delete a node"<<endl;
    cout<<"9-Delete after given node"<<endl;
    cout<<"10-Exit"<<endl;
```

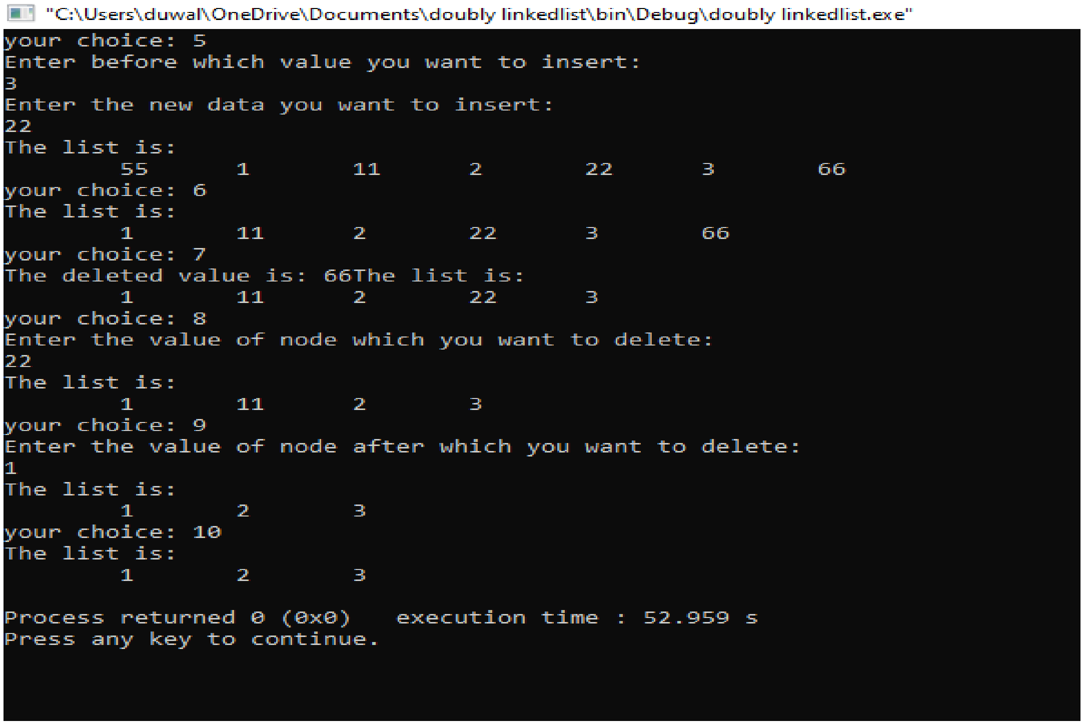
```
while(choice!=10){
    cout<<"your choice: ";
    cin>>choice;
    switch (choice){
    case 1:
        creation();
        break;
    case 2:
        insert_begin();
        break;
    case 3:
        insert_end();
        break;
    case 4:
        insert_afternode();
        break;
    case 5:
        insert_beforenode();
        break;
    case 6:
        del_begin();
        break;
    case 7:
        del_end();
        break;
    case 8:
        del_node();
        break;
        case 9:
            del_after();
            break;
        }
        display_list();
    }
}
```

Output:



"C:\Users\duwal\OneDrive\Documents\doubly linkedlist\bin\Debug\doubly linkedlist.exe"

```
1-Creating a new list
2-Inserting at beginning
3-Inserting at the end
4-Inserting after given node
5-Inserting before given node
6-Delete beginning node
7-Delete ending node
8-Delete a node
9-Delete after given node
10-Exit
your choice: 1
Enter the data for the list(insert -1 to end the list): 1
Enter the data to be stored at the end:         2
Enter the data to be stored at the end:         3
Enter the data to be stored at the end:         -1
The list is:
        1       2       3
your choice: 2
Enter the data to be inserted at the beginning
55
The list is:
        55      1       2       3
your choice: 3
Enter the data to be stored at the end:         66
The list is:
        55      1       2       3       66
your choice: 4
Enter after which value you want to insert:
1
Enter the new data you want to insert:
11
The list is:
        55      1       11      2       3       66
your choice: 5
Enter before which value you want to insert:
3
Enter the new data you want to insert:
22
The list is:
        55      1       11      2       22      3       66
your choice: 6
The list is:
        1       11      2       22      3       66
your choice: 7
The deleted value is: 66The list is:
        1       11      2       22      3
```


"C:\Users\duwal\OneDrive\Documents\doubly linkedlist\bin\Debug\doubly linkedlist.exe"

```
your choice: 5
Enter before which value you want to insert:
3
Enter the new data you want to insert:
22
The list is:
        55      1       11      2       22      3       66
your choice: 6
The list is:
        1       11      2       22      3       66
your choice: 7
The deleted value is: 66The list is:
        1       11      2       22      3
your choice: 8
Enter the value of node which you want to delete:
22
The list is:
        1       11      2       3
your choice: 9
Enter the value of node after which you want to delete:
1
The list is:
        1       2       3
your choice: 10
The list is:
        1       2       3

Process returned 0 (0x0)   execution time : 52.959 s
Press any key to continue.
```

Conclusion:

In this way doubly linked list was implemented. By using above code doubly linked list can be create , delete ,insert as per user required.

4.4 WAP to implement priority queue using linked list

Problem analysis:

The problem here is to implement priority queue using linked list. Where we have to create a queue according to its priority and dequeue as FIFO principle according to its priority order.

Algorithm :

```
Enqueue
    Step 1 : Allocate memory for the new node
    Step 2 : SET NEW_NODE=>DATA=VAL
    Step 3 : SET NEW_NODE=>NEXT=NULL
    Step 4 : PTR= START
    Step 5 : Repeat Step while NEW_NODE=>DATA!=NULL
        Step 5.1: IF NEW_NODE=>PRIORITY<START=>PRIORITY
                SET NEW_NODE=>START
                SET START=NEW_NODE
            ELSE
                Repeat  While PTR=>NEXT!=NULL AND
                PTR=>NEXT=>PRIORITY<NEW_NODE=>PRIORITY
                    PTR=PTR=>NEXT
                [END OF WHILE LOOP]
                SET NEW_NODE=>NEXT=PTR=>NEXT
                SET PTR=>NEXT = NEW_NODE
    Step 6 : EXIT
Dequeue
    Step 1 : SET PTR=START=>NEXT
    Step 2 : FREE START
    Step 3 : SET START = PTR
    Step 4 : EXIT
```

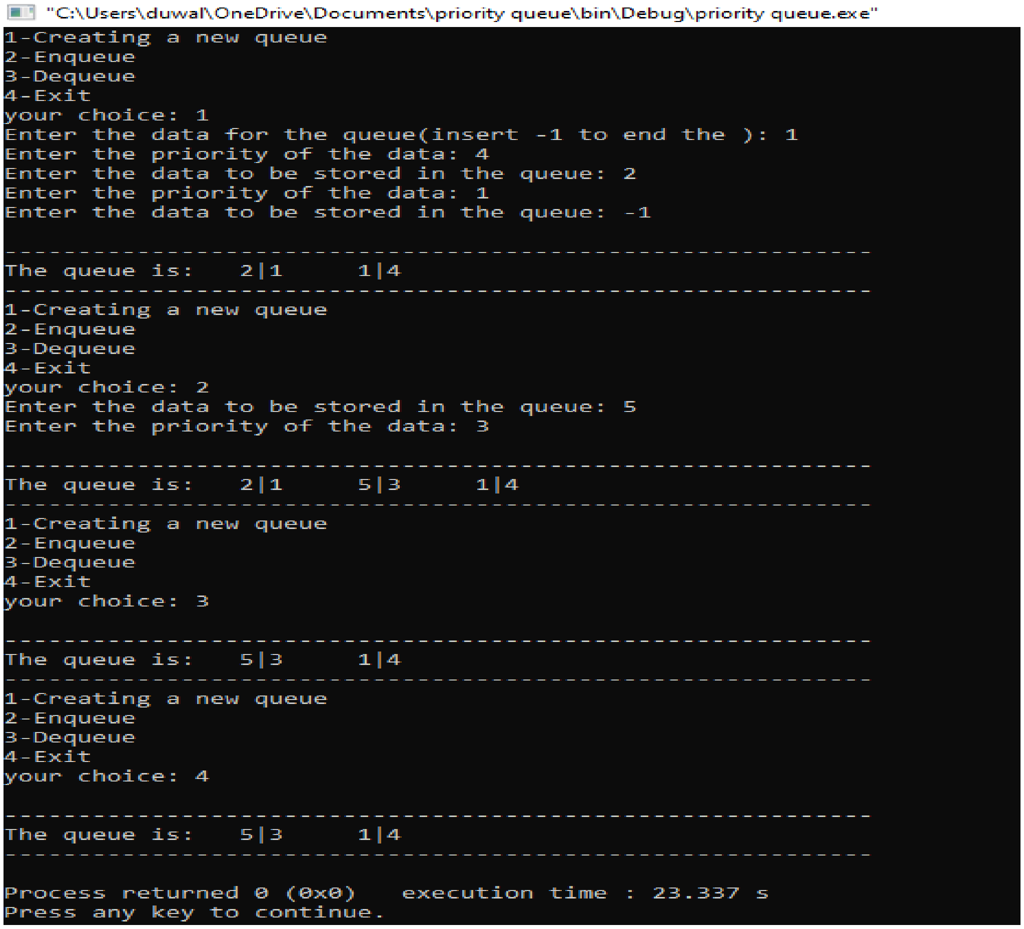
Source Code:

```
#include<iostream>
using namespace std;
class Queue{
    struct node
    {
        int data;
        int priority;
        struct node * next;
    };
   public:
    struct node * start;
    struct node * newnode,* temp,* ptr;
    void creation()
    {
            newnode = new node;
            cout<<"Enter the data for the queue(insert -1 to end the ): ";
            cin>>newnode->data;
            cout<<"Enter the priority of the data: ";
            cin>>newnode->priority;
            newnode->next=NULL;
            if (start==NULL)
            {
                start=newnode;
                temp=newnode;
            }
            else
            {
```

```
            temp->next=newnode;
            temp=newnode;
        }
    do{
            enqueue();
    }while (newnode->data!=-1);
}
void enqueue()
{
    newnode=new node;
    cout<<"Enter the data to be stored in the queue: ";
    cin>>newnode->data;
    newnode->next=NULL;
    if (newnode->data!=-1)
    {
        cout<<"Enter the priority of the data: ";
        cin>>newnode->priority;
        ptr=start;
        if (newnode->priority<start->priority)
        {
            newnode->next=start;
            start=newnode;
        }
        else
        {
            while(ptr->next!=NULL && ptr->next->priority<newnode->priority )
            {
                ptr=ptr->next;
                }
                newnode->next=ptr->next;
                ptr->next=newnode;
            }
        }
    }
    void dequeue()
    {
        ptr=start->next;
        delete start;
        start=ptr;
    }
    void display_queue()
    {
        ptr=start;
        cout<<endl;
        cout<<"----------------------------------------------------------"<<endl;
        cout<<"The queue is: \t";
        cout<<ptr->data<<"|"<<ptr->priority;
        while(ptr->next!=NULL)
        {
            ptr=ptr->next;
            cout<<"\t"<<ptr->data<<"|"<<ptr->priority;
        }
        cout<<endl;
        cout<<"----------------------------------------------------------"<<endl;
    }
};
```

```
int main()
{
    class Queue q;
    q.start=NULL;
    int choice;
    while(choice!=4){
        cout<<"1-Creating a new queue "<<endl;
        cout<<"2-Enqueue "<<endl;
        cout<<"3-Dequeue "<<endl;
        cout<<"4-Exit "<<endl;
        cout<<"your choice: ";
        cin>>choice;
        switch (choice){
        case 1:
            q.creation();
            break;
        case 2:
            q.enqueue();
            break;
        case 3:
            q.dequeue();
            break;
        }
        q.display_queue();
    }
}
```

Output:



```
"C:\Users\duwal\OneDrive\Documents\priority queue\bin\Debug\priority queue.exe"
1-Creating a new queue
2-Enqueue
3-Dequeue
4-Exit
your choice: 1
Enter the data for the queue(insert -1 to end the ): 1
Enter the priority of the data: 4
Enter the data to be stored in the queue: 2
Enter the priority of the data: 1
Enter the data to be stored in the queue: -1

-----------------------------------------------------------
The queue is:   2|1     1|4
-----------------------------------------------------------
1-Creating a new queue
2-Enqueue
3-Dequeue
4-Exit
your choice: 2
Enter the data to be stored in the queue: 5
Enter the priority of the data: 3

-----------------------------------------------------------
The queue is:   2|1     5|3     1|4
-----------------------------------------------------------
1-Creating a new queue
2-Enqueue
3-Dequeue
4-Exit
your choice: 3

-----------------------------------------------------------
The queue is:   5|3     1|4
-----------------------------------------------------------
1-Creating a new queue
2-Enqueue
3-Dequeue
4-Exit
your choice: 4

-----------------------------------------------------------
The queue is:   5|3     1|4
-----------------------------------------------------------

Process returned 0 (0x0)   execution time : 23.337 s
Press any key to continue.
```

Conclusion:

In this way priority queue was implemented. By using above code we can arrange the queue insertion and deletion according to its priority

4.5 WAP to implement STACK using linked list

Problem Analysis:

The problem here is to use linked list to create a stack. Using linked list we can insert and delete data, so we use it to use it as a stack by implementing LIFO principle. We push data to the linked list and pop data from the linked list.

Algorithm:

```
Push
Step 1 : Allocate memory for the newnode and named it as NEW_NODE
Step 2 : SET NEW_NODE=>DATA = VAL
Step 3 : IF TOP =NULL
                SET NEW_NODE=>NEXT=NULL
                SET TOP=NEW_NODE
        ELSE
                SET NEW_NODE=>NEXT = TOP
                SET TOP = NEW_NODE
        [ END OF IF ]
Step 4 : END
Pop
Step 1 : IF TOP=NULL
                PRINT "UNDERFLOW"
                Goto Step 5
        [END OF IF ]
Step 2 : SET PTR=TOP
Step 3 : SET TOP = TOP=>NEXT
Step 4 : FREE PTR
Step 5 : END
```

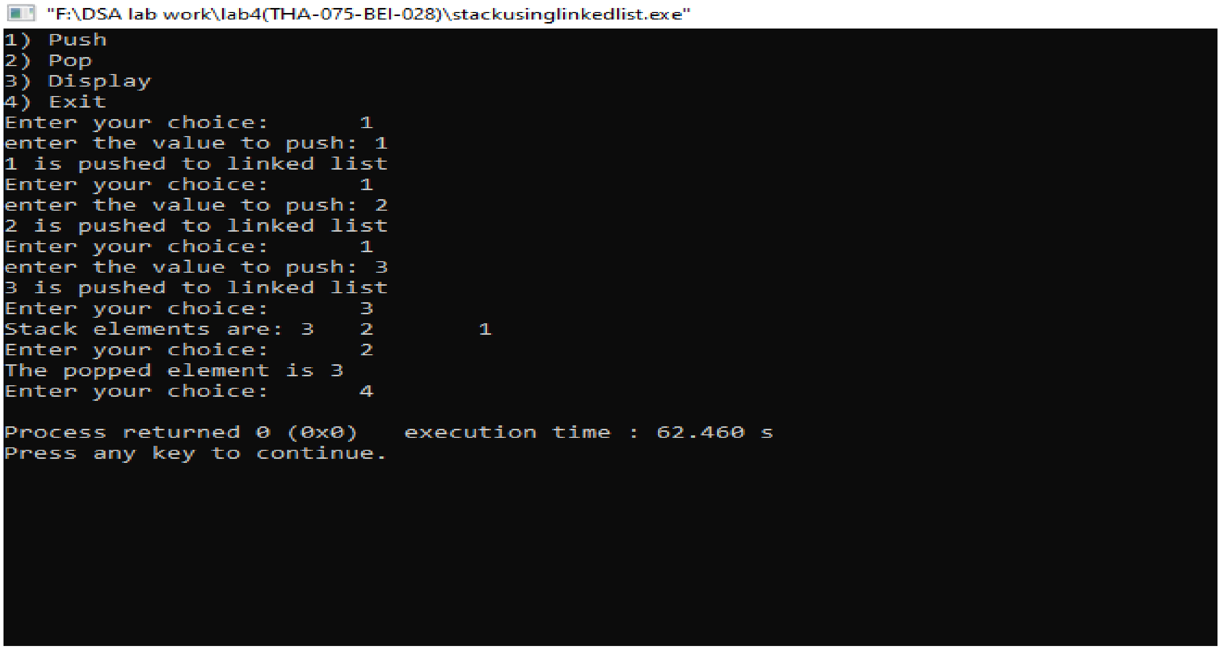
SourceCode:

```
//WAP to implement STACK using linked list
#include <iostream>
using namespace std;
struct Node
{
  int data;
  struct Node *next;
};
struct Node* head = NULL;
void push()
{
    int val;
      cout<<"enter the value to push: ";
      cin>>val;
      Node *newnode = new Node;
      newnode->data = val;
      cout<<newnode->data<<" is pushed to linked list "<<endl;
      if(head == NULL)
      {
         head = newnode;
         head->next = NULL;
      }
      else
      {
         newnode->next = head;
         head = newnode;
      }
```

```
}
void pop()
{

    if(head==NULL)
    cout<<"Stack Underflow"<<endl;
    else
    {
        Node *tmp = head;
        cout<<"The popped element is "<< head->data <<endl;
        head = head->next;
        delete tmp;
    }
}
void display()
{
   struct Node* t;
   if(head==NULL)
   cout<<"Stack is not created yet."<<endl;
   else
    {
        t = head;
        cout<<"Stack elements are: ";
        while (t!=NULL)
        {
            cout<< t->data <<"\t";
            t = t->next;
        }
   }
  cout<<endl;
}
int main() {
  int ch, val;
    cout<<"1) Push"<<endl;
    cout<<"2) Pop"<<endl;
    cout<<"3) Display"<<endl;
    cout<<"4) Exit"<<endl;
    do {
    cout<<"Enter your choice: \t";
    cin>>ch;
    switch(ch)
     {
      case 1:
        {
          push();
          break;
        }

      case 2:
        {
          pop();
          break;
        }
      case 3:
        {
          display();
```

```
            break;
        }
      case 4:
        {
            break;
        }
      default:
        {
            cout<<"Invalid input"<<endl;
        }
    }
  }while(ch!=4);
}
```

Output:



"F:\DSA lab work\lab4(THA-075-BEI-028)\stackusinglinkedlist.exe"

```
1) Push
2) Pop
3) Display
4) Exit
Enter your choice:       1
enter the value to push: 1
1 is pushed to linked list
Enter your choice:       1
enter the value to push: 2
2 is pushed to linked list
Enter your choice:       1
enter the value to push: 3
3 is pushed to linked list
Enter your choice:       3
Stack elements are: 3    2       1
Enter your choice:       2
The popped element is 3
Enter your choice:       4

Process returned 0 (0x0)   execution time : 62.460 s
Press any key to continue.
```

Conclusion:

In this way stack was implemented using linked list . By using above code we can push and pop the node in linked list according to LIFO principle.

4.6 WAP to implement QUEUE using linked list

Problem analysis:

The problem here is to implement the principle of FIFO principle to implement queue using linked list data structure. For this, we insert and delete the data from the rear end and the front respectively. We define two node pointers, *front and *rear which points to front end and rear end respectively. To insert data we create a new node and initialize the next pointer of rear to the newly created node and initialize rear end with the new node. For deleting data, we first store the front pointer to a temporary variable and initialize front with the value of next pointer and then delete the temp variable.

Algorithm :

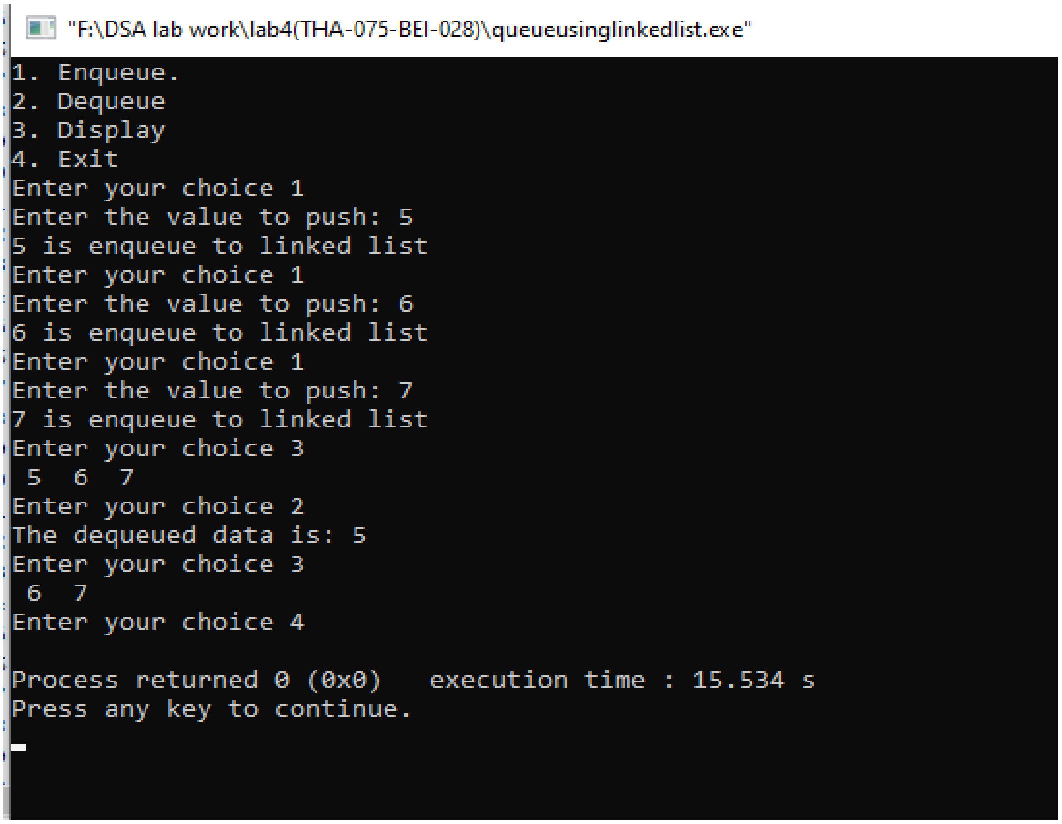
```
Enqueue:
    Step 1 : Allocate memory for the new node and name it as PTR
    Step 2 : SET PTR=>DATA =VAL
    Step 3 : IF FRONT = NULL
                SET FRONT = REAR=PTR
                SET FRONT => NEXT = REAR => NEXT = NULL
             ELSE
                SET REAR=>NEXT = PTR
                SET REAR=PTR
                SET REAR=>NEXT = NULL
             [END OF IF]
    Step 4: END
Dequeue :
    Step 1 : IF FRONT = NULL
                Write "UNDERFLOW"
                Goto Step 5
             [END OF IF]
    Step 2 : SET PTR = FRONT
    Step 3 : SET FRONT = FRONT=>NEXT
    Step 4 : FREE PTR
    Step 5 : END
```

Sourcecode:

```
//WAP to implement QUEUE using linked list
#include<iostream>
using namespace std;
struct node
{
    int data;
    node* next;
};
struct node* head = NULL;
void enqueue()
{
    int val;
    cout<<"Enter the value to push: ";
    cin>>val;
    node *ptr = head;
    node *newnode = new node;
    newnode->data = val;
    cout<<newnode->data<<" is enqueue to linked list "<<endl;
    if(head == NULL)
    {
        head= newnode;
        newnode->next = NULL;
    }
    else
    {
        while(ptr->next != NULL)
        {
            ptr = ptr->next;
        }
        newnode->next = ptr->next;
        ptr->next = newnode;
    }
}
void dequeue()
{
    if(head == NULL)
    {
        cout<<"Underflow!!"<<endl;
    }
    else
    {
        node *ptr = head;
        cout<<"The dequeued data is: "<<head->data<<endl;
        head = head->next;
        delete ptr;
    }
}
void display()
{
    if(head == NULL)
    {
        cout<<"\nThe list is empty!!"<<endl;
    }
    else
    {
        node *ptr = head;
        while(ptr != NULL)
        {
            cout<<" "<<ptr->data<<" ";
            ptr = ptr->next;
        }
        cout<<endl;
    }
}
int main()
{
    int ch;
        cout<<"1. Enqueue."<<endl;
        cout<<"2. Dequeue"<<endl;
        cout<<"3. Display"<<endl;
        cout<<"4. Exit"<<endl;
        do
        {
        cout<<"Enter your choice ";
        cin>>ch;
        switch (ch)
        {
        case 1:
        {
            enqueue();
            break;
        }
        case 2:
```

```
        {
            dequeue();
            break;
        }
        case 3:
        {
            display();
            break;
        }
        case 4:
            break;

        default :
        {
            cout<<"Invalid input";
            break;
        }
        }
    } while (ch != 4);

    return 0;
}
```

Output:


"F:\DSA lab work\lab4(THA-075-BEI-028)\queueusinglinkedlist.exe"

```
1. Enqueue.
2. Dequeue
3. Display
4. Exit
Enter your choice 1
Enter the value to push: 5
5 is enqueue to linked list
Enter your choice 1
Enter the value to push: 6
6 is enqueue to linked list
Enter your choice 1
Enter the value to push: 7
7 is enqueue to linked list
Enter your choice 3
 5  6  7
Enter your choice 2
The dequeued data is: 5
Enter your choice 3
 6  7
Enter your choice 4

Process returned 0 (0x0)   execution time : 15.534 s
Press any key to continue.
```

Conclusion :

In this way queue was implemented using linked list . By using above code we can insert and delete the node in linked list according to FIFO principle.

4.7 WAP to store a polynomial using linked list. Also perform addition and subtraction on two polynomials

Problem analysis:

In this program we perform an addition and subtraction of two polynomials by storing a polynomial using linked list.

Algorithm :

```
Step 1: Create a structure for coefficient , exponents and next for
        new node.
Step 2: Call a function to create a polynomial, add a polynomial,
        Subtract a polynomial, and display a polynomial.
Step 3: Scan two polynomials node by node comparing for the
        exponents.
Step 4: If the exponents are equal, add/subtract their coefficients.
Step 5: If the exponents are not equal, simply add/subtract a node
        in the new list.
```

Sourcecode:

```
#include<iostream>
#include<cstdlib>
#include<cmath>

using namespace std;
struct node
{
   int check;
   int info,xp;
   node *next;
};
class POLY
```

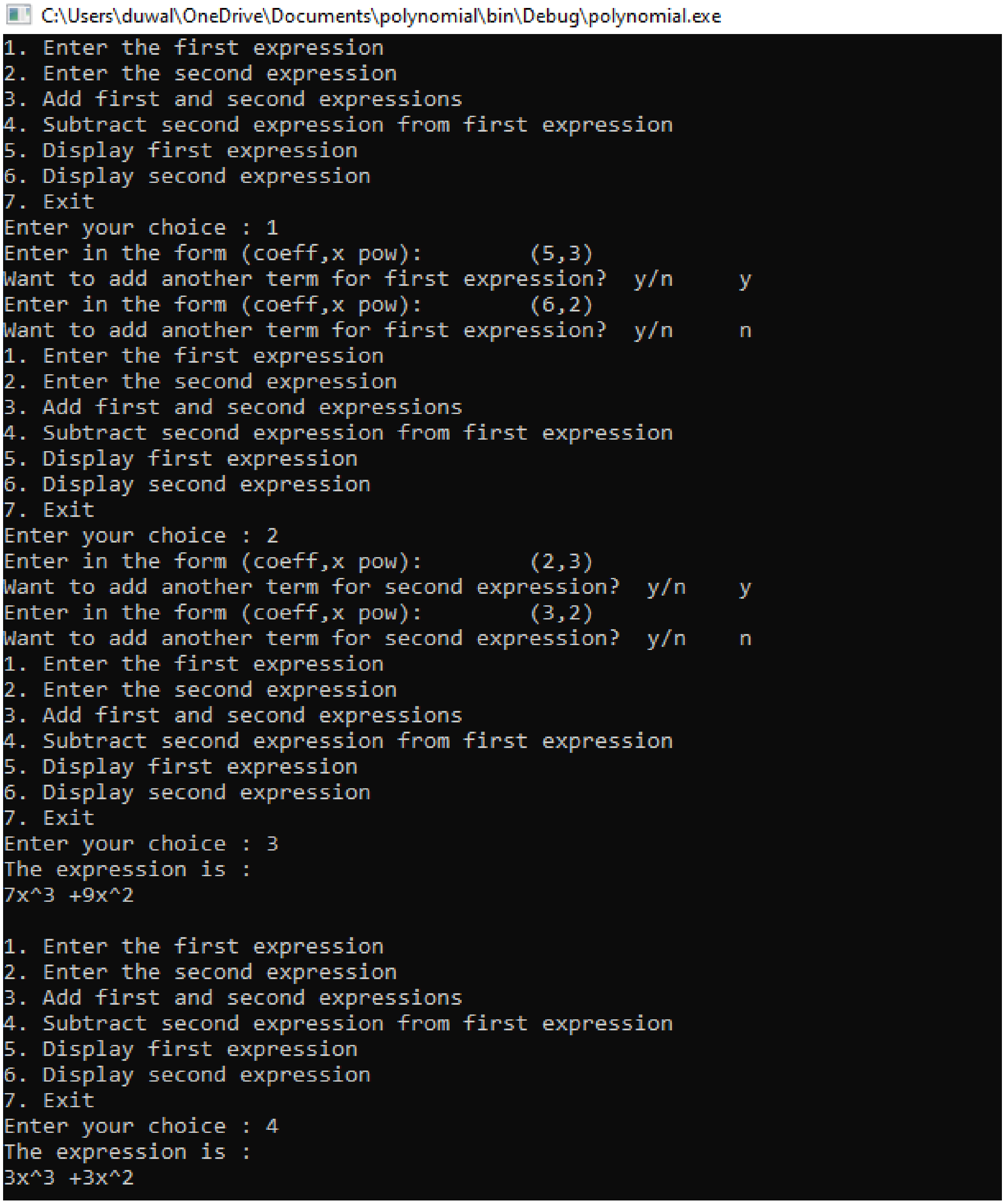
```
{
   node *START;
public:
   POLY():START(NULL){}
   void AddExpression(int,int);
   POLY operator + (POLY &);
   POLY operator - (POLY &);
   bool DisplayExpression();
};
void POLY::AddExpression(int num,int x)
{
   node *temp=new node;
   if(temp==NULL)
      cout<<"Failed to initialize the memory for new block.\n";
   else
      {
       temp->info=num;
       temp->xp=x;
       temp->next=NULL;
       if(START==NULL)
         START=temp;
       else
       {
          node *ptr;
          ptr=START;
          while(ptr->next!=NULL)
            ptr=ptr->next;
          ptr->next=temp;
          }
        }
}
POLY POLY::operator+(POLY &second)
{
    POLY t;
    if(START==NULL)
        {
            cout<<"There is no first polynomial expression.\n";
            return t;
        }
    else if(second.START==NULL)
        {
            cout<<"There is no second polynomial expression.\n";
            return t;
        }
    else
    {
        int c;
        node *p1,*p2;
        p1=START;
        while(p1!=NULL)
        {
            c=0;
            p2=second.START;
            while(p2!=NULL)
            {
                if(p1->xp==p2->xp)
        {
          c=1;
          p2->check=1;
          t.AddExpression((p1->info+p2->info),p1->xp);
          break;
        }
        p2=p2->next;
      }
      if(c==0)
        t.AddExpression(p1->info,p1->xp);
      p1=p1->next;

    }
    p2=second.START;
    while(p2!=NULL)
    {
      if(p2->check!=1)
        t.AddExpression(p2->info,p2->xp);
      p2=p2->next;
    }
  return t;
  }
}
POLY POLY::operator-(POLY &second)
{
  POLY t;
  if(START==NULL)
    {
        cout<<"There is no first polynomial expression.\n";
        return t;
    }
else if(second.START==NULL)
    {
        cout<<"There is no second polynomial expression.\n";
        return t;
    }
else
{
    int c;
    node *p1,*p2;
    p1=START;
    while(p1!=NULL)
    {
        c=0;
        p2=second.START;
        while(p2!=NULL)
        {
            if(p1->xp==p2->xp)
            {
                c=1;
                p2->check=1;
                t.AddExpression((p1->info-p2->info),p1->xp);
                break;
            }
            p2=p2->next;
        }
```

```
            if(c==0)
                t.AddExpression(p1->info,p1->xp);
            p1=p1->next;

        }
        p2=second.START;
        while(p2!=NULL)
        {
            if(p2->check!=1)
                t.AddExpression(-p2->info,p2->xp);
            p2=p2->next;
        }
     return t;
    }
}
bool POLY::DisplayExpression()
{
    if(START==NULL)
        {
            cout<<"No expression\n";
            return false;
        }
    else
    {
     node *ptr;
     ptr=START;
     cout<<"The expression is :\n";
     while(ptr!=NULL)
```

```
        {
            if(ptr==START &&ptr->info>=0)
                cout<<ptr->info<<"x^"<<ptr->xp<<" ";
            else if(ptr->info>=0)
             cout<<"+"<<ptr->info<<"x^"<<ptr->xp<<" ";
            else
             cout<<ptr->info<<"x^"<<ptr->xp<<" ";
            ptr=ptr->next;
        }
     cout<<"\n\n";
     return true;
    }
}
int main()
{
  POLY e1,e2,e3;
  int choice,info,x,y,z;
  char ch;
  while(1)
  {
      cout<<"1. Enter the first expression\n2. Enter the second expression\n3. Add first and second expressions\n4. Subtract second expression from first expression\n5. Display first expression\n6. Display second expression\n7. Exit\nEnter your choice : ";
      cin>>choice;
      switch(choice)
      {
      case 1:
        {
               char c='y';
                while(c=='y'||c=='Y')
                   {
                     cout<<"Enter in the form (coeff,x pow):   \t";
                     cin>>ch>>info>>ch>>x>>ch;
                     e1.AddExpression(info,x);
                     cout<<"Want to add another term for first expression?  y/n\t";
                     cin>>c;
                   }
                    break;
        }
    case 2:
        {
               char c='y';
                while(c=='y'||c=='Y')
                   {
                     cout<<"Enter in the form (coeff,x pow):   \t";
                     cin>>ch>>info>>ch>>x>>ch;
                     e2.AddExpression(info,x);
                     cout<<"Want to add another term for second expression?  y/n\t";
                     cin>>c;
                   }
                    break;
        }
    case 3:
        {


            e3=e1+e2;
            bool b=e3.DisplayExpression();
```

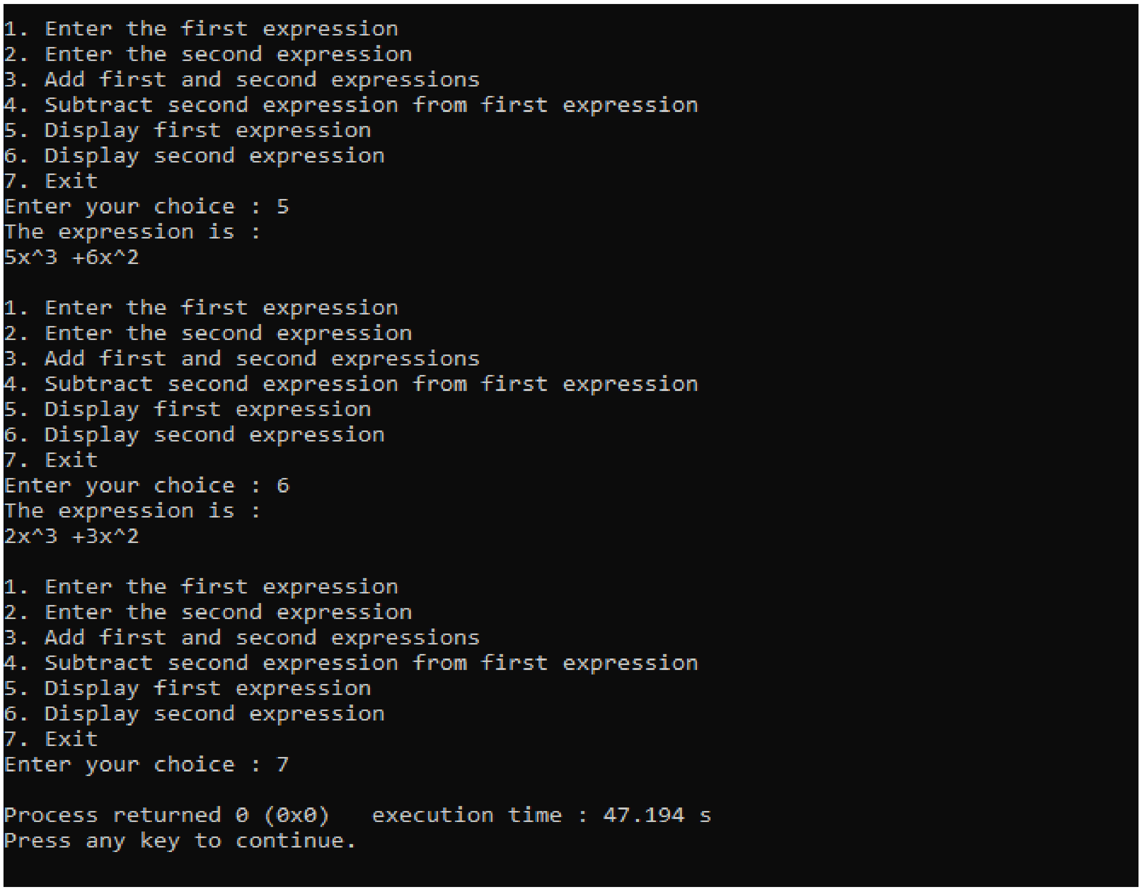
```
          break;
       }
    case 4:
       {

          e3=e1-e2;
         bool b=e3.DisplayExpression();
          break;
       }
    case 5:
       {
          bool b=e1.DisplayExpression();
          break;
       }
    case 6:
       {
          bool b=e2.DisplayExpression();
          break;
       }
    default :exit(0);

      }

  }
  return 0;
}
```

Output :



C:\Users\duwal\OneDrive\Documents\polynomial\bin\Debug\polynomial.exe

```
1. Enter the first expression
2. Enter the second expression
3. Add first and second expressions
4. Subtract second expression from first expression
5. Display first expression
6. Display second expression
7. Exit
Enter your choice : 1
Enter in the form (coeff,x pow):        (5,3)
Want to add another term for first expression?  y/n     y
Enter in the form (coeff,x pow):        (6,2)
Want to add another term for first expression?  y/n     n
1. Enter the first expression
2. Enter the second expression
3. Add first and second expressions
4. Subtract second expression from first expression
5. Display first expression
6. Display second expression
7. Exit
Enter your choice : 2
Enter in the form (coeff,x pow):        (2,3)
Want to add another term for second expression?  y/n    y
Enter in the form (coeff,x pow):        (3,2)
Want to add another term for second expression?  y/n    n
1. Enter the first expression
2. Enter the second expression
3. Add first and second expressions
4. Subtract second expression from first expression
5. Display first expression
6. Display second expression
7. Exit
Enter your choice : 3
The expression is :
7x^3 +9x^2

1. Enter the first expression
2. Enter the second expression
3. Add first and second expressions
4. Subtract second expression from first expression
5. Display first expression
6. Display second expression
7. Exit
Enter your choice : 4
The expression is :
3x^3 +3x^2
```



```
1. Enter the first expression
2. Enter the second expression
3. Add first and second expressions
4. Subtract second expression from first expression
5. Display first expression
6. Display second expression
7. Exit
Enter your choice : 5
The expression is :
5x^3 +6x^2

1. Enter the first expression
2. Enter the second expression
3. Add first and second expressions
4. Subtract second expression from first expression
5. Display first expression
6. Display second expression
7. Exit
Enter your choice : 6
The expression is :
2x^3 +3x^2

1. Enter the first expression
2. Enter the second expression
3. Add first and second expressions
4. Subtract second expression from first expression
5. Display first expression
6. Display second expression
7. Exit
Enter your choice : 7

Process returned 0 (0x0)   execution time : 47.194 s
Press any key to continue.
```

Conclusion : Thus the program for performing addition and subtraction on two polynomials was performed successfully.

LAB 5: RECURSION

5.1. Write a recursive program to find factorial of a given number

Problem analysis :

Here, the problem is to find out the factorial of a number using recursion. A recursive function is a self calling function. What that means is that a recursive function will call its own function inside its main body. A factorial of a number is the number obtained after multiplying the given number with number less than the current number until it reaches 1. The factorial of 0 and 1 is equal to 1. Here in the program, we first check to see if the given number is zero or 1. If so, we return the value 1 but if the number is not 0 or 1 or negative we call the function recursively and return the product of the number and the value returned from the recursive function with parameter 1 less than the number. i.e. num * factorial(num-1)

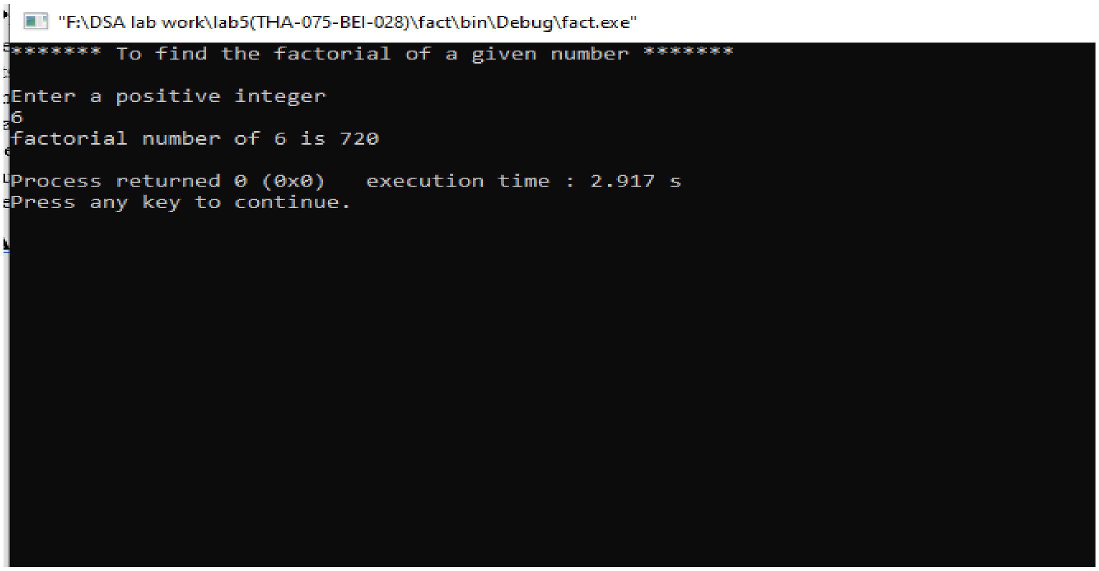
Algorithm :

```
Step 1 : Create a recursion function fact having one parameter num
        i.e fact(num)
Step 2 : If num = 1 or num = 0
                Return 1 ;
        Else
                Return num*fact(num-1)
        [End of if]
Step 3 : End
```

Sourcecode:

```
//Write a recursive program to find factorial of a given number
#include <iostream>
using namespace std;
int fact(int n)
{
   if(n==1)
      return 1;
   else
      return(n*fact(n-1));
}
int main()
{
  int n;
  cout<<"******* To find the factorial of a given number *******"<<endl<<endl;
  cout<<"Enter a positive integer "<<endl;
  cin>>n;
  cout<<"factorial number of "<<n<<" is "<<fact(n)<<endl;
}
```

Output :



"F:\DSA lab work\lab5(THA-075-BEI-028)\fact\bin\Debug\fact.exe"

```
******* To find the factorial of a given number *******

Enter a positive integer
6
factorial number of 6 is 720

Process returned 0 (0x0)   execution time : 2.917 s
Press any key to continue.
```

Conclusion :

In this way we can find the factorial of the positive integer given by user using recursive function.

5.2. Write a recursive program to find N terms Fibonacci series

Problem analysis :

Fibonacci series is a type of sequence such that each number is the sum of the two preceding ones, starting from 0 and 1. That is, F_0 = 0 and F_1 = 1, and

$F_n = F_{n-1} + F_{n-2}$, for n > 1. In this program we first check if the number of terms is 0 or 1, if it is, we return the num. Else we call the recursive function and return the sum of the functions having parameters $(n-1)|_{f(n-1)}$ and $(n-2)|_{f(n-2)}$.

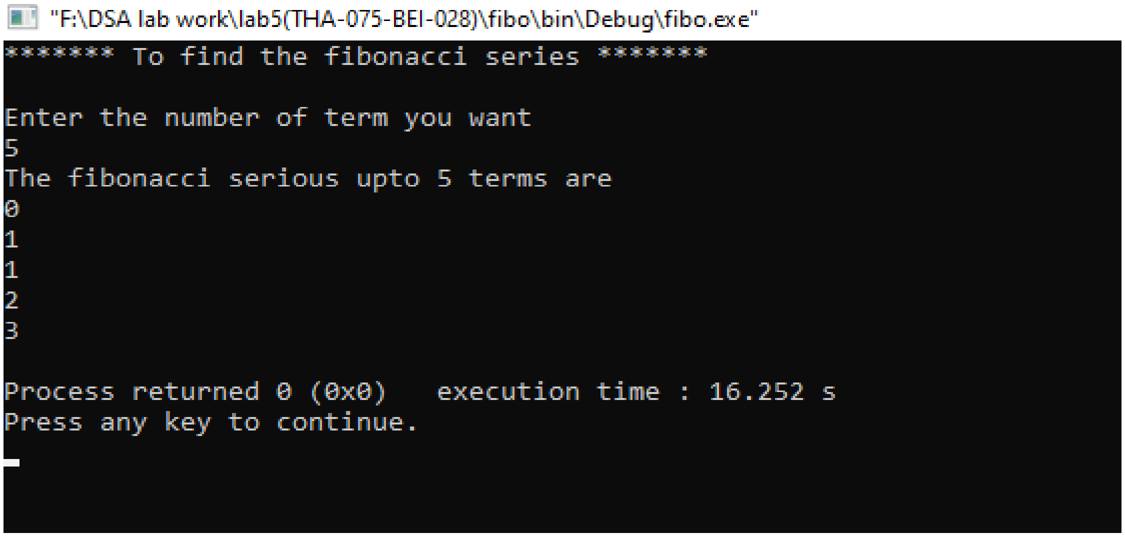
Algorithm:

```
Step 1 : Create a recursive function fibo having one parameter num
            i.e fibo(num)
Step 2 : If num = 1
                        Return 0
            Else if num = 2
                        Return 1
            Else
                        Return fibo(num-2)+fibo(num-1)
Step 3 : End
```

Sourcecode:

```
// Write a recursive program to find N terms Fibonacci series
#include <iostream>
using namespace std;
int fibo(int n)
{
   if(n==1)
      return 0;
   else if (n==2)
      return 1;
   else
      return (fibo(n-2)+fibo(n-1));
}
int main()
{
   int n;
   cout<<"******* To find the fibonacci series *******"<<endl<<endl;
   cout << "Enter the number of term you want "<<endl;
   cin>>n;
   cout<<"The fibonacci serious upto "<<n <<" terms are " <<endl;
   for(int i=1;i<=n;i++)
   {
      cout<<fibo(i)<<endl;
   }
}
```

Output :



"F:\DSA lab work\lab5(THA-075-BEI-028)\fibo\bin\Debug\fibo.exe"

```
******* To find the fibonacci series *******

Enter the number of term you want
5
The fibonacci serious upto 5 terms are
0
1
1
2
3

Process returned 0 (0x0)   execution time : 16.252 s
Press any key to continue.
```

Conclusion :

In this way we can find the Fibonacci series using recursive function upto the number of terms given by user.

5.3. Write a recursive program to solve Tower of Hanoi

Problem analysis :

Tower of Hanoi is a mathematical puzzle where we have three rods and n disks. The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

1) Only one disk can be moved at a time.

2) Only a smaller disk can be placed above a bigger disk, the opposite is strictly forbidden.

The problem here is that, we have to solve that puzzle using recursion though we cannot solve it by looping . So, this problem is typically recursion based.

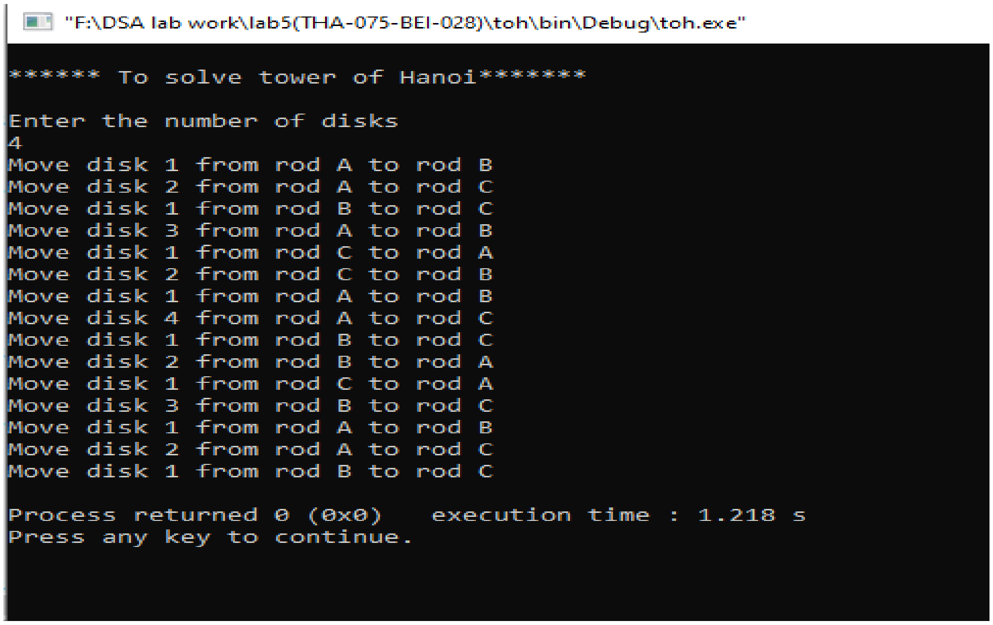
Algorithm :

```
Step 1 : Create a recursive function Toh having four parameters n ,source ,dest ,aux
                I,e Toh(n, source , dest ,aux)
Step 2: If n=1
        Move disk 1 from source to dest
        Else
                Toh(n-1, source, aux, dest)
                Move n from source to dest
                Toh(n-1, aux , dest , source)
        [End of if ]
Step 3: End
```

Sourcecode:

```
//Write a recursive program to solve Tower of Hanoi.
#include <iostream>
using namespace std;
void towerOfHanoi(int n, char from_rod,char to_rod, char aux_rod)
{
        if (n == 1)
        {
                cout << "Move disk 1 from rod " << from_rod <<" to rod " << to_rod<<endl;
                return;
        }
        towerOfHanoi(n - 1, from_rod, aux_rod, to_rod);
        cout << "Move disk " << n << " from rod " << from_rod <<" to rod " << to_rod << endl;
        towerOfHanoi(n - 1, aux_rod, to_rod, from_rod);
}
int main()
{
        int n;
        cout<<endl<<"****** To solve tower of Hanoi*******  "<<endl<<endl;
        cout<<"Enter the number of disks"<<endl;
        cin>>n;
        towerOfHanoi(n, 'A', 'C', 'B'); //A,B,C are rods
        return 0;
}
```

Output :


"F:\DSA lab work\lab5(THA-075-BEI-028)\toh\bin\Debug\toh.exe"

```
****** To solve tower of Hanoi*******

Enter the number of disks
4
Move disk 1 from rod A to rod B
Move disk 2 from rod A to rod C
Move disk 1 from rod B to rod C
Move disk 3 from rod A to rod B
Move disk 1 from rod C to rod A
Move disk 2 from rod C to rod B
Move disk 1 from rod A to rod B
Move disk 4 from rod A to rod C
Move disk 1 from rod B to rod C
Move disk 2 from rod B to rod A
Move disk 1 from rod C to rod A
Move disk 3 from rod B to rod C
Move disk 1 from rod A to rod B
Move disk 2 from rod A to rod C
Move disk 1 from rod B to rod C

Process returned 0 (0x0)   execution time : 1.218 s
Press any key to continue.
```

Conclusion:

In this way by using recursive function we can easily solve the n number of disks puzzle of tower of hanoi

5.4. Write a recursive program to find Greatest Common Division GCD of two numbers.

Problem analysis :

The greatest common divisor of two numbers is the largest number that divides both the numbers. We find out GCD of two numbers with a recursive function. First of all, we check if one of the number is 0, if it is, then the second number is the GCD of the two numbers. we calculate the remainder and the quotient of two numbers after division. And then we call the recursive function and pass the second number and the remainder as function parameters which will recursively check the conditions until one of them is zero.

Algorithm :

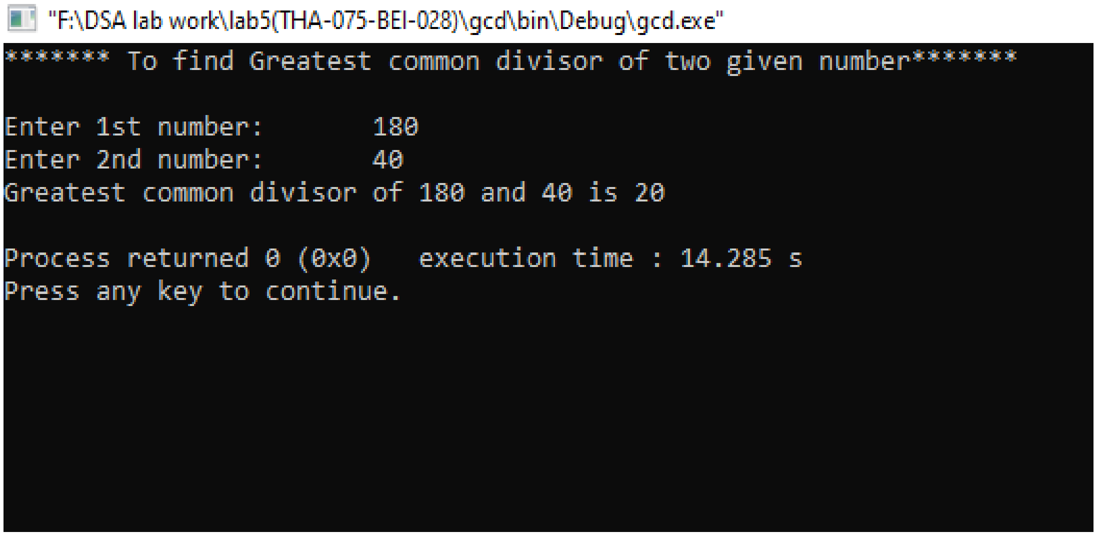
```
greatestCommonDivisor(n1, n2)
Step 1:
        If n2 = 0:
                gcd = n1
        Else:
                Set quotient = n1 / n2
                Set remainder = n1 % n2
Step 2:
        Call greatestCommonDivisor(n2, remainder)

Step 3: EXIT
```

Sourcecode:

```
#include <iostream>
int gcd(int n1,int n2)
{
   if(n2!=0)
      return gcd(n2,n1%n2);
   else
      return n1;
}
using namespace std;

int main()
{
   int n1,n2;
   cout<<"******* To find Greatest common divisor of two given
number*******"<<endl<<endl;
   cout<<"Enter 1st number: \t";
   cin>>n1;
   cout<<"Enter 2nd number: \t";
   cin>>n2;
```

```
    cout<<"Greatest common divisor of "<<n1<<" and " <<n2 <<" is "<<gcd(n1,n2)<<endl;
}
```

Output :



"F:\DSA lab work\lab5(THA-075-BEI-028)\gcd\bin\Debug\gcd.exe"

```
******* To find Greatest common divisor of two given number*******

Enter 1st number:       180
Enter 2nd number:       40
Greatest common divisor of 180 and 40 is 20

Process returned 0 (0x0)   execution time : 14.285 s
Press any key to continue.
```

Conclusion :

In this way we can find the greatest common division of the two numbers given by user.

LAB 6: Trees

TITLE : Write a menu driven program for the following operations on Binary Search Tree(BST) of integers

i. Create a BST of N integers: 5, 10, 25, 2, 8, 15, 24, 14, 7, 8, 35, 2
ii. Traverse the BST in Inorder , Preorder and Postorder
iii. Search the BST for a given element(KEY) and print the appropriate message
iv. Exit

PROGRAM ANALYSIS :

In this program , we create a binary search tree by inserting a given integer and perform the different operations like traversing in preorder , inorder and postorder and searching the BST element.

Algorithm :

Preorder

```
    Step 1 : Repeat Steps 2 to 4 While Tree!=NULL
    Step 2 : Write TREE=>DATA
    Step 3 : PREORDER(TREE=>LEFT)
    Step 4 : PREORDER(TREE=>RIGHT)
            [END OF LOOP]
    Step 5 : END
```

Inorder

```
    Step 1 : Repeat Steps 2 to 4 While Tree!=NULL
    Step 2 : INORDER(TREE=>LEFT)
    Step 3 : Write Tree=>DATA
    Step 4 : INORDER(Tree=>RIGHT)
            [ END OF LOOP ]
    Step 5 : END
```

Postorder

```
    Step 1 : Repeat Steps 2 to 4 While Tree!=NULL
```

```
Step 2 : POSTORDER(TREE=>LEFT)
Step 3 :POSTORDER(TREE=>RIGHT)
Step 4 : Write TREE=>DATA
        [End of loop]
Step 5 : End
```

Sourcecode:

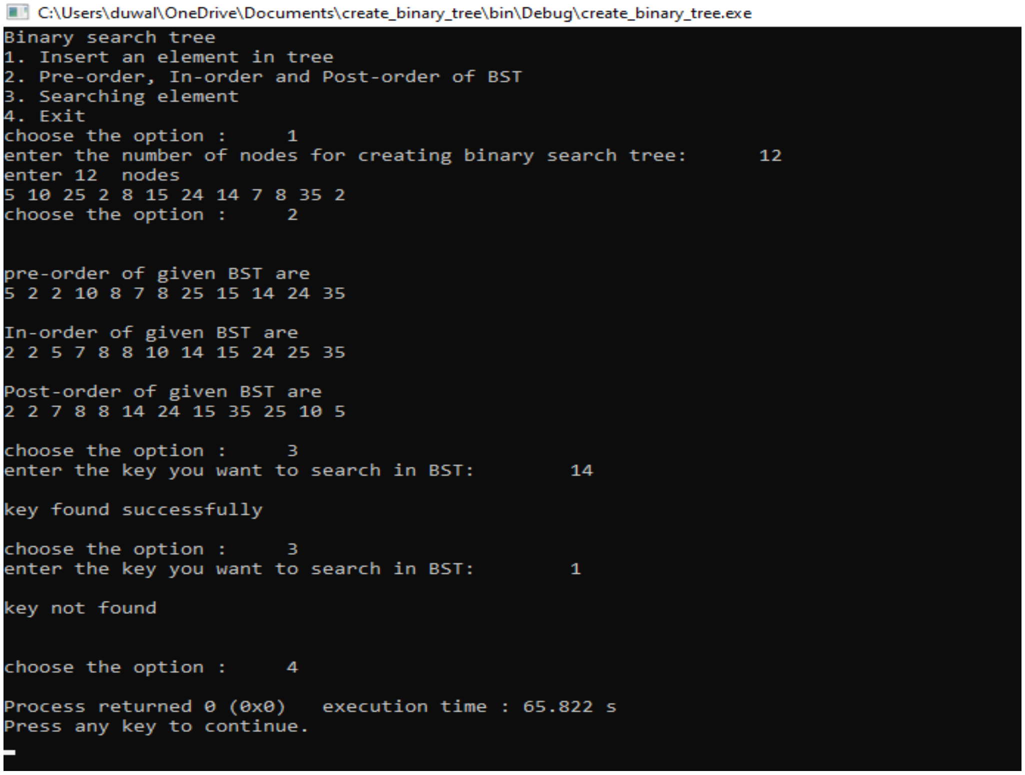
```
#include <iostream>
using namespace std;
struct node{
   struct node *left;
   int data;
   struct node *right;
};
struct node *newNode(int val)
{
   struct node *temp =  new struct node();
   temp->data = val;
   temp->left = temp->right = NULL;
   return temp;
}
struct node* insert(struct node* node, int val)
{
   if (node == NULL) return newNode(val);
   if (val < node->data)
      node->left  = insert(node->left, val);
   else
       node->right = insert(node->right, val);
   return node;
}
void inorder(struct node* node)
{
   if(node!=NULL)//LNR
   {
      inorder(node->left);
      cout<<node->data<<" ";
      inorder(node->right);
   }
}
void preorder(struct node* node)
{
   if(node!=NULL)//NLR
   {
      cout<<node->data<<" ";
      preorder(node->left);
      preorder(node->right);
   }
}
void postorder(struct node* node)
{
   if(node!=NULL)//LRN
   {
      postorder(node->left);
      postorder(node->right);
      cout<<node->data<<" ";
   }
}
```

```
void search(struct node* node ,int val)
{
   if(node!=NULL)
      {
       if(node->data==val)
           cout<<endl<<"key found successfully"<<endl<<endl;
       else
           if(val<node->data)
               search(node->left,val);
           else
               search(node->right,val);
      }
   else
       cout<<endl<<"key not found"<<endl<<endl<<endl;
}
int main()
{
 int n,val,ch;
 struct node* new_node=NULL;
 cout<<"Binary search tree"<<endl;
 cout<<"1. Insert an element in tree"<<endl;
 cout<<"2. Pre-order, In-order and Post-order of BST"<<endl;
 cout<<"3. Searching element "<<endl;
 cout<<"4. Exit"<<endl;
 do{
 cout<<"choose the option :\t";
 cin>>ch;
 switch(ch){
   case 1:  cout<<"enter the number of nodes for creating binary search tree:\t";
            cin>> n;
            cout<<"enter "<<n<<"  nodes"<<endl;
            for(int i=0;i<n;i++)
            {
                cin>>val;
                new_node=insert(new_node,val);
            }
            break;
   case 2: cout<<endl<<endl;
           cout<<"pre-order of given BST are "<<endl;
           preorder(new_node);
           cout<<endl<<endl;
           cout<<"In-order of given BST are "<<endl;
           inorder(new_node);
           cout<<endl<<endl;
           cout<<"Post-order of given BST are "<<endl;
           postorder(new_node);
           cout<<endl<<endl;
           break;
   case 3: cout<<"enter the key you want to search in BST:\t";
           cin>>val;
           search(new_node,val);
           break;
   case 4: break;
   default: cout<<"invalid input"<<endl;
          }
}while(ch!=4);
```

return 0;

}

Output:



C:\Users\duwal\OneDrive\Documents\create_binary_tree\bin\Debug\create_binary_tree.exe

```
Binary search tree
1. Insert an element in tree
2. Pre-order, In-order and Post-order of BST
3. Searching element
4. Exit
choose the option :     1
enter the number of nodes for creating binary search tree:      12
enter 12  nodes
5 10 25 2 8 15 24 14 7 8 35 2
choose the option :     2


pre-order of given BST are
5 2 2 10 8 7 8 25 15 14 24 35

In-order of given BST are
2 2 5 7 8 8 10 14 15 24 25 35

Post-order of given BST are
2 2 7 8 8 14 24 15 35 25 10 5

choose the option :     3
enter the key you want to search in BST:        14

key found successfully

choose the option :     3
enter the key you want to search in BST:        1

key not found


choose the option :     4

Process returned 0 (0x0)   execution time : 65.822 s
Press any key to continue.
```

Conclusion :

In this way we create a BST of given n integers number and traverse them in preorder , inorder and postorder and we can also search the BST for given element .If the element is in BST then it displays key found successfully message otherwise it displays key not found message.

LAB 7 : Sorting

7.1 WAP to implement Insertion Sorting Algorithm

Problem analysis :

The problem here is to sort the array values using insertion sorting algorithm. First of all the array of values to be sorted is divided into two sets. One that stores sorted values and another that contains unsorted values. The sorting algorithm will proceed until there are elements in the unsorted set.

Algorithm :

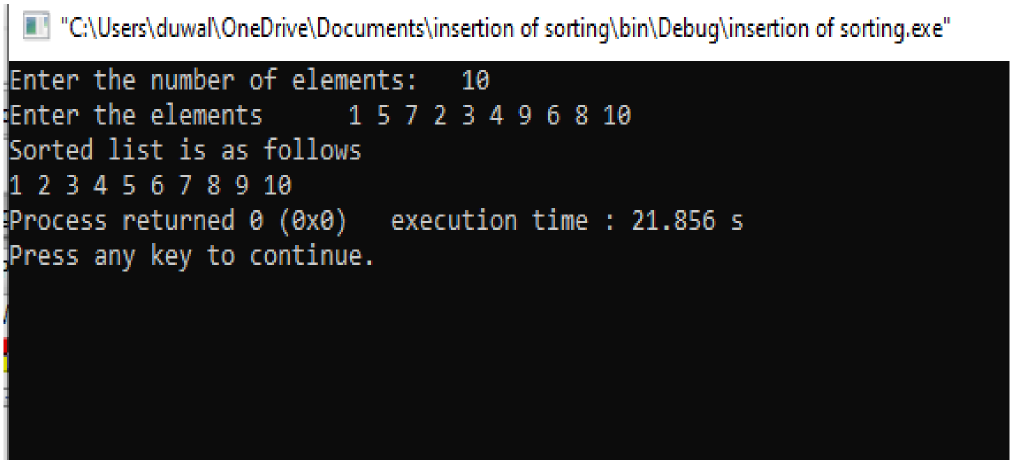
```
INSERTING-SORT (ARR,N)

Step 1 : Repeat Steps 2 to 5 for k = 1 to N-1

Step 2 : SET TEMP=ARR[K]

Step 3 : SET J=K-1

Step 4 : Repeat while TEMP<=ARR[J]

                    SET ARR[J+1]=ARR[J]

                    SET J = J-1

          [END OF INNER LOOP]

Step 5 : SET ARR[J+1]=TEMP

          [END OF LOOP]

Step 6 : EXIT
```

Sourcecode:

```
#include<iostream>

using namespace std;

int main()

{

   int i,j,n,temp,a[30];

   cout<<"Enter the number of elements:";

   cin>>n;

   cout<<"Enter the elements\t";
```

```
    for(i=0;i<n;i++)

    {
        cin>>a[i];
    }
    for(i=1;i<=n-1;i++)
    {
        temp=a[i];
        j=i-1;

        while((temp<a[j])&&(j>=0))
        {
            a[j+1]=a[j];    //moves element forward
            j=j-1;
        }
        a[j+1]=temp;    //insert element in proper place
    }
    cout<<"Sorted list is as follows\n";
    for(i=0;i<n;i++)
    {
        cout<<a[i]<<" ";
    }
    return 0;
}
```

Output :



"C:\Users\duwal\OneDrive\Documents\insertion of sorting\bin\Debug\insertion of sorting.exe"

```
Enter the number of elements:   10
Enter the elements      1 5 7 2 3 4 9 6 8 10
Sorted list is as follows
1 2 3 4 5 6 7 8 9 10
Process returned 0 (0x0)   execution time : 21.856 s
Press any key to continue.
```

Conclusion :

In this way we can sort the value of array using insertion sorting algorithm.

7.2 WAP to implement Merge Sorting Algorithm

Problem analysis :

The problem here is to sort the value of array using merge sorting algorithm. If the array is of length 0 or 1, then it is already sorted. Otherwise, divide the unsorted array into two sub-arrays of about half the size. Use merge sort algorithm recursively to sort each sub-array and Merge the two sub-arrays to form a single sorted list.

Algorithm:

```
MERGE (ARR,BEG,MID,END)
Step 1 : [INITIALIZE] SET I = BEG , J=MID+1 , INDEX=0
Step 2 : Repeat while (I<=MID) AND (J<=END)
                IF ARR[I] < ARR[J]
                        SET TEMP[INDEX]=ARR[I]
                        SET I = I+1
                ELSE
                        SET TEMP[INDEX]=ARR[J]
                        SET J = J+1
                [END OF IF]
                SET INDEX=INDEX+1
        [END OF LOOP]
Step 3 : [Copy the remaining elements of right sub-array,if any]
                IF I > MID
                        Repeat while J<=END
                                SET TEMP[INDEX]=ARR[J]
                                SET INDEX=INDEX+1, SET J = J+1
                        [END OF LOOP]
                [Copy the remaining elements of left sub-array, if any]
                ELSE
                        Repeat while I<=END
                                SET TEMP[INDEX]=ARR[I]
                                SET INDEX=INDEX+1, SET I=I+1
```

```
                    [END OF LOOP]
            [END OF IF]
    Step 4 : [Copy the contents of TEMP back to ARR] SET K=0
    Step 5 : Repeat while K<INDEX
                    SET ARR[K]=TEMP[K]
                    SET K=K+1
            [END OF LOOP]
    Step 6 : END

    MERGE_SORT(ARR , BEG, END)
    Step 1 : IF BEG<END
                    SET MID = (BEG+END)/2
                    CALL MERGE_SORT (ARR , BIG , MID)
                    CALL MERGE_SORT (ARR, MID+1 , END)
                    MERGE ( ARR , BEG , MID, END)
            [END OF IF]
    Step 2 : END
```

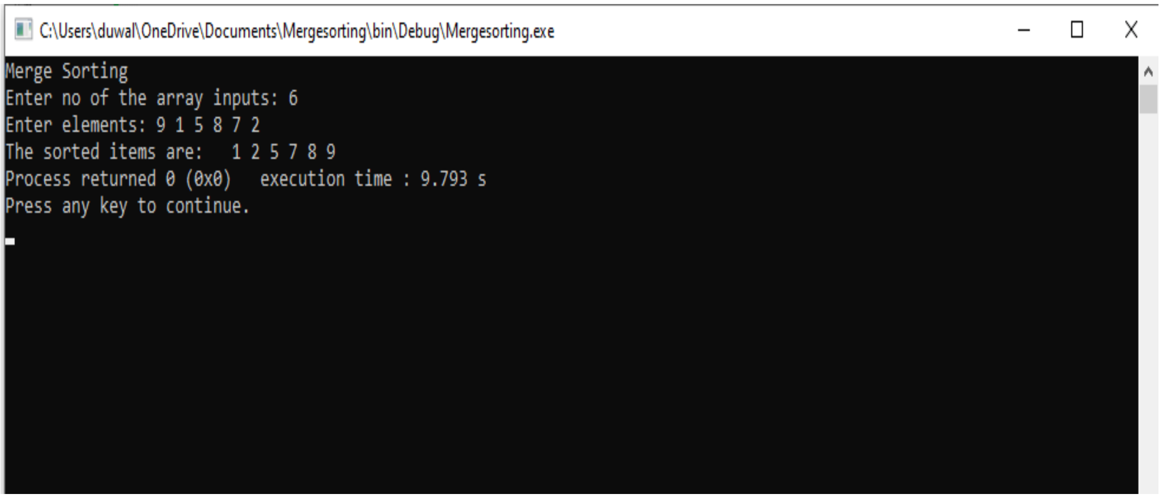
Sourcecode:

```
#include<iostream>
using namespace std;
void merge(int A[],int beg,int mid,int end){
   int i=beg;
   int j=mid+1;
   int index=beg;
   int temp[end+1],k;
   while(i<=mid && j<=end){
      if(A[i]<A[j]){
         temp[index]=A[i];
```

```
            i++;
        }
        else{
            temp[index]=A[j];
            j++;
        }
        index++;
    }
    if(i>mid){
        while(j<=end){
            temp[index]=A[j];
            index++;
            j++;
        }
    }
    else{
        while(i<=mid){
            temp[index]=A[i];
            index++;
            i++;
        }
    }
    k=beg;
    while(k<index){
        A[k]=temp[k];
        k++;
    }
}
```

```
void merge_sort(int A[], int beg,int end){
    int mid;
    if(beg<end){
        mid=(beg+end)/2;
        merge_sort(A,beg,mid);
        merge_sort(A,mid+1,end);
        merge(A,beg,mid,end);
    }
}
int main(){
    int n;
    cout<<"Merge Sorting\n";
    cout<<"Enter no of the array inputs: ";
    cin>>n;
    int arr[n];
    cout<<"Enter elements:\t";
    for(int i=0;i<n;i++)
        cin>>arr[i];
    merge_sort(arr,0,n-1);
    cout<<"The sorted items are: \t";
    for(int i=0;i<n;i++)
        cout<<arr[i]<<" ";
    return 0;
}
```

Output:



C:\Users\duwal\OneDrive\Documents\Mergesorting\bin\Debug\Mergesorting.exe

```
Merge Sorting
Enter no of the array inputs: 6
Enter elements: 9 1 5 8 7 2
The sorted items are:   1 2 5 7 8 9
Process returned 0 (0x0)   execution time : 9.793 s
Press any key to continue.
```

Conclusion:

In this way we can sort the value of array using merge sorting algorithm.