ASSIGNMENT NO. 02
NAME : LAUKIK BHAGAWAN BHOGALE
ROLL NO. 14
SUB : DSA
CLASS : SE IT
BATCH A

FILE NAME :stack.cpp

```
#include<iostream>

using namespace std;

struct node
{
 int info;
 struct node *next;
};

 class stack
 {
   node *top;
   public:

    stack();
    void push(int);
    int pop();
    int isempty();
    int topdata();
 };

  stack::stack()
  {
   top=NULL;
  }

  void stack::push(int x)
  {
    node *p;
    p=new node;
    p->info=x;
    p->next=top;
    top=p;
  }

  int stack::pop()
  {
   node *temp;
   temp=top;
   int q=temp->info;
   top=top->next;
   delete temp;
   return q;
  }

  int stack::isempty()
  {
```

```
   if(top==NULL)
    return 1;
   else
    return 0;
  }

  int stack::topdata()
  {
   return top->info;
  }
```

FILE NAME : assignment2.cpp

```
#include <iostream>
#include "stack.cpp"
#include<cstring>
#include<math.h>

class expre
{
char inf[20],pre[20],post[20];

  public:
       void inftopre();
       void inftopost();
       void strreve(char a[]);
       int isp(char);
       int icp(char);
       void preeval();
       void posteval();
       float eval(char,int,int);
};

void expre::strreve(char a[])
{
  char temp;
  int i=0;
  int j=strlen(a)-1;
  while(i<j)
  {
  temp=a[i];
  a[i]=a[j];
  a[j]=temp;
  i++;
  j--;
  }
}
int expre::icp(char x)
{
 int q;
 switch(x)
 {
     case'+':q=1;
     break;
     case'-':q=1;
     break;
```

```
    case'*':q=2;
    break;
    case'/':q=2;
    break;
    case'%':q=3;
    break;
    case'^':q=4;
    break;
    case'(':q=5;
    break;
 }
 return q;
}

int expre::isp(char x)
{
 int q;
 switch(x)
 {
    case'+':q=1;
    break;
    case'-':q=1;
    break;
    case'*':q=2;
    break;
    case'/':q=2;
    break;
    case'%':q=3;
    break;
    case'^':q=4;
    break;
    case'(':q=5;
    break;
 }
 return q;
}


void expre::inftopre()
{
int j=0;
stack s;
cout<<"enter the infix expression"<<endl;
cin>>inf;
strreve(inf);
  for(int i=0;i<strlen(inf);i++)
  {
  if(inf[i]=='(')
  {
  inf[i]=')';
  }
  else if(inf[i]==')')
  {
   inf[i]='(';
  }
  }
  for(int i=0;i<strlen(inf);i++)
```

```
  {
   char x=inf[i];
   if(isalnum(x))
   {
   pre[j++]=x;
   }
   else if(x=='(')
   {
   s.push(x);
   }
   else if(x==')')
   {
   while ((x=s.pop())!='(')
   {
   pre[j++]=s.pop();
   }
   }
   else
   {
    while(!(s.isempty()) && icp(x)<isp(s.topdata()))
    {
      pre[j++]=s.pop();
    }
    s.push(x);
   }
   }

  while(!(s.isempty()))
  {
  pre[j++]=s.pop();
}
pre[j]='\0';
strreve(pre);
cout<<"The Prefix expression is: "<<pre<<endl;
}

void expre::inftopost()
{
 int j=0;
 stack s;
 cout<<"enter the infix expression"<<endl;
 cin>>inf;
  for(int i=0;i<strlen(inf);i++)
  {
   char x=inf[i];
   if(isalnum(x))
   {
   post[j++]=x;
   }
   else if(x=='(')
   {
   s.push(x);
   }
   else if(x==')')
   {
   while ((x=s.pop())!='(')
   {
```

```
   post[j++]=s.pop();
    }
    }
    else
    {
     while(!(s.isempty()) && icp(x)<isp(s.topdata()))
     {
       post[j++]=s.pop();
     }
     s.push(x);
    }
  }
   while(!(s.isempty()))
   {
   post[j++]=s.pop();
 }
 pre[j]='\0';
 cout<<"The Postfix expression is: "<<post<<endl;
  }

   float expre::eval(char x,int x1,int x2)
   {
    float a;
    switch(x)
    {
     case'+':return(x1+x2);
     case'-':return(x1-x2);
     case'*':return(x1*x2);
     case'/':return(x1/x2);
     case'%':return(x1%x2);
     case'^':return(pow(x1,x2));
    }
   }

  void expre::preeval()
  {
   stack s;
   float r;
   cout<<"Enter Prefix Expression :"<<endl;
   cin>>pre;
   strreve(pre);
   for(int i=0;pre[i]!='\0';i++)
   {
    char x=pre[i];
    if(isalpha(x))
    {
     cout<<"Enter value for"<<char(x)<<endl;
     cin>>x;
     s.push(x);
    }
    else if(isdigit(x))
    {
     s.push(x-48);
    }
    else
    {
     int s1=s.pop();
```

```
    int s2=s.pop();
    r=eval(x,s1,s2);
    s.push(r);
   }
   }
   r=s.pop();
   cout<<r;
  }

  void expre::posteval()
 {
  stack s;
  float r;
  cout<<"Enter postfix Expression :"<<endl;
  cin>>post;
  for(int i=0;post[i]!='\0';i++)
  {
   char x=post[i];
   if(isalpha(x))
   {
    cout<<"Enter value for"<<char(x)<<endl;
    cin>>x;
    s.push(x);
   }
   else if(isdigit(x))
   {
    s.push(x-48);
   }
   else
   {
    int s2=s.pop();
    int s1=s.pop();
    r=eval(x,s1,s2);
    s.push(r);
   }
   }
   r=s.pop();
   cout<<r;
  }



int main()
{
expre e;
int ch;
while(1)
{
cout<<"*********MENU********"<<endl;
cout<<"\n1.infix to prefix \n2.infix to postfix \n3. Postfix Evaluation \n4. Prefix evaluation \n5. exit"
<<endl;
cin>>ch;
switch(ch)
{
 case 1:e.inftopre();
    break;
```

```
  case 2:e.inftopost();
    break;

  case 3:
  e.posteval();
  break;

  case 4:
  e.preeval();
  break;

  case 5:
    exit(0);
}
}
}
```

*******OUTPUT*******

```
 *********MENU********

1.infix to prefix
2.infix to postfix
3. Postfix Evaluation
4. Prefix evaluation
5. exit
1
enter the infix expression
A/B^C+D*E-A*C
The Prefix expression is: -+/A^BC*DE*AC
*********MENU********

1.infix to prefix
2.infix to postfix
3. Postfix Evaluation
4. Prefix evaluation
5. exit
2
enter the infix expression
A/B^C+D*E-A*C
The Postfix expression is: ABC^/DE*AC*-+
*********MENU********

1.infix to prefix
2.infix to postfix
3. Postfix Evaluation
4. Prefix evaluation
5. exit
3
Enter postfix Expression :
```

23*21-/53*+
21*********MENU********

1.infix to prefix
2.infix to postfix
3. Postfix Evaluation
4. Prefix evaluation
5. exit
4
Enter Prefix Expression :
+*2/3-21*53
21*********MENU********

1.infix to prefix
2.infix to postfix
3. Postfix Evaluation
4. Prefix evaluation
5. exit
5