```
//design and implement a menu driven program for expression conversion frm inf to post, post
to pre and evalutaion of post using stack
#include<iostream>
#include<string>
#include<bits/stdc++.h>
#define size 1000
using namespace std;

class Stack
{
	public:
		string stack[size];
		int top;

		Stack()
		{
			top=-1;
		}

		bool isFull()
		{
			if (top==size-1)
			{
				return true;
			}
			else
			{
				return false;
			}
		}

		bool isEmpty()
		{
			if(top==-1)
			{
				return 1;
			}
			else
			{
				return 0;
			}
		}

		void push(string s)
		{
			if(isFull())
			{
				cout<<"\nStack is Full"<<endl;
			}
			else
```

```
				{
					top+=1;
					stack[top]=s;
				}
			}

			string pop()
			{
				string temp;
				if(isEmpty())
				{
					cout<<"\nStack is Empty"<<endl;
				}
				else
				{
					temp=stack[top];
					top-=1;
					return temp;
				}
			}
};

class expression
{
	public:
		string post,pre,in;
		Stack s;

		bool isOperator(char x)
		{
			if (x=='+' || x=='-' || x=='*' || x=='/' || x=='^')
			{
				return true;
			}
			else
			{
				return false;
			}
		}

		int prec(string op)
		{
			if (op=="+" || op=="-")
			{
				return 1;
			}
			else if (op=="*" || op=="/")
			{
				return 2;
			}
		else if (op=="^")
		{
			return 3;
		}
		else
		{
			return 0;
		}
	}

	void in_to_post()
	{
		cout<<"\nEnter the infix expression:";
		cin>>in;

		for (int i=0;i<in.length();i++)
		{
			if ((in[i]>='a' && in[i]<='z') || (in[i]>='A' && in[i]<='Z'))
			{
				post+=in[i];
			}
			else if (in[i]=='(')
			{
				s.push("(");
			}
			else if (in[i]==')')
			{
				while ((!s.isEmpty()) && s.stack[s.top]!="(")
				{
					string t=s.stack[s.top];
					s.pop();
					post+=t;
				}
				if (s.stack[s.top]=="(")
				{
					s.pop();
				}
			}
			else
			{
				while ((!s.isEmpty()) &&
prec(string(1,in[i]))<=prec(s.stack[s.top]))
				{
					string t=s.stack[s.top];
					s.pop();
					post+=t;
				}
				s.push(string(1,in[i]));
			}
		}
```

```
                    while(!s.isEmpty())
                    {
                            string t=s.stack[s.top];
                            s.pop();
                            post+=t;
                    }
                    cout<<"\t\nInfix Expression:";
                    cout<<in;
                    cout<<"\t\nPostfix Expression:";
                    cout<<post<<endl;
            }

            void pre_to_in()
            {
                    cout<<"\nEnter the prefix expression:";
                    cin>>pre;
          int n = pre.length();
          for (int i = n - 1; i >= 0; i--)

{
if (isOperator(pre[i]))
{
string op1 = s.pop();
string op2 = s.pop();
string s1 = "(" + op1 + pre[i] + op2 + ")";
s.push(s1);
}
else
{
s.push(string(1, pre[i]));
}
}
in = s.pop();
cout << "Converted infix expression: " << in << endl;
}

            void post_eva()
            {
                    cout<<"\nEnter the postfix expression to evaluate:";
                    cin>>post;

                    cout<<"\t\nPostfix Expression:";
                    cout<<post;
                    cout<<"\t\nAnswer:";

                    for (int i=0;i<post.length();i++)
                    {
                            if (isOperator(post[i]))
                            {
                                    int op_1,op_2,ans;
                                    string op2=s.pop();
                                    string op1=s.pop();

                                    stringstream stm1(op1);
                                    stringstream stm2(op2);

                                    stm1>>op_1;
                                    stm2>>op_2;

                                    if(post[i]=='+')
                                    {
                                        ans=op_1+op_2;
                                    }
                                    else if(post[i]=='-')
                                    {
                                        ans=op_1-op_2;
                                    }
                                    else if(post[i]=='*')
                                    {
                                        ans=op_1*op_2;
                                    }
                                    else if(post[i]=='/')
                                    {
                                        ans=op_1/op_2;
                                    }
                                    else if(post[i]=='^')
                                    {
                                        ans=pow(op_1,op_2);
                                    }

                                    stringstream stm3;
                                    stm3<<ans;
                                    string ans1=stm3.str();
                                    s.push(ans1);

                            }
                            else
                            {
                                    if(isdigit(post[i]))
                                    {
                                        s.push(string(1,post[i]));
                                    }
                            }
                    }
                    cout<<s.pop()<<endl;
            }
};

int main()
{
    expression ob1;
    int ch;
    do
    {
            cout<<"\n1.Prefix to Infix conversion\n2.Infix to Postfix conversion\n3.Postfix
Evaluation\n4.Exit\nEnter choice:";
            cin>>ch;

            switch(ch)
            {
                    case 1:
                            ob1.pre_to_in();
                            break;
                    case 2:
                            ob1.in_to_post();
                            break;
                    case 3:
                            ob1.post_eva();
                            break;
                    case 4:
                    break;
                    default:
                    cout << "Invalid choice!\n";
                    break;
            }
    }while(ch!=4);
}
```