EXPRESSION CONVERSION

ROLL NUMBER : 2002
BATCH : E-10

MAIN

```
#include <iostream>
#include<string>
#include "stack.h"
#include "inpost.h"
#include "inpre.h"
#include <stdexcept>
#include "stack_int.h"
#include<stdlib.h>
using namespace std;

int isoperator(char x)
{
    if( ( (x== '+') || (x == '-') || (x == '*') || (x == '/') || (x == '^') ) )
        return 1;

    else
        return 0;
}

int read(string s)
{
    int len=s.length();

    for(int i=0;i<len-1;i++)
    {
        if(s[i]=='(')
        {
            if (! (s[i+1] != '(' || !isalnum(s[i+1]) ))
            {
                cout<<"Invalid !!!\n\n";
                return 0;
            }
        }

        else if( isalnum(s[i]) )
        {
            if( !  (s[i+1] == ')' || isoperator(s[i+1]) ) )
            {
                cout<<"Invalid !!!\n\n";
                return 0;
            }

        }

        else if( isoperator(s[i]) )
```

```
        {
            if (! (s[i+1] == '(' || isalnum(s[i+1]) ))
            {
                cout<<"Invalid !!!\n\n";
                return 0;
            }
        }

        else if( s[i]==')')
        {
            if(! (isoperator(s[i+1]) || s[i+1]==')') )
                return 0;
        }
    }

    return 1;

}

int main()
{
        stacks s;
        stack_int p;
        inpost post;
        inpre pre;

        string arr,arr1;
        string ans;

        do
        {
                int ch;
                cout<<"\n\tMENU\n";

                cout<<"\t\t1.Enter Expression\n\t\t2.Infix To Postfix\n\t\t"
                                "3.Infix To Prefix\n\t\t4.Postfix Evaluation\n\t\t"
                                "5.Prefix Evaluation\n\t\t6.Re_Enter\n\t\t7.Exit\n";

                cout<<"Enter Choice : ";
                cin>>ch;
                cout<<endl;

                switch(ch)
                {
                                        case 1:
                                        {
                                                cout<<"\n\t\tEnter String in infix form :";
                                                cin>>arr;
                                                cout<<endl;

                                                int check= read(arr);
```

```
                while(check==0)
                {
                        cout<<"\n\t\tEnter String in infix form :";
                        cin>>arr;
                        cout<<endl;

                       check=read(arr);
                 }

                break;
        }

        case 2:
        {
                cout<<"\n\t\tPostfix Version  of Given Expression is : ";
                post.infix_to_postfix(arr,s);
                break;
        }

        case 3:
        {
                cout<<"\n\t\tPrefix Version  of Given Expression is : ";
                pre.infix_to_prefix(arr,s);
                break;
        }

        case 4:
        {
                cout<<"\n\t\tEnter String in postfix Form(With Single
Digits as operands) :";
                cin>>arr1;
                cout<<endl;

                cout<<"\n\t\tAnswer of Given Expression is :
"<<post.postfix_eval(arr1,p)<<endl;
                break;
        }

        case 5:
        {
                cout<<"\n\t\tEnter String in prefix Form(With Single
Digits as operands) :";
                cin>>arr1;
                cout<<endl;

                cout<<"\n\t\tAnswer of Given Expression is :
"<<pre.prefix_eval(arr1,p)<<endl;
                break;
        }

        case 6:
        {
```

```
                                        main();
                                        break;
                                }

                                case 7:
                                {
                                        exit(0);
                                        break;
                                }
                        }
                }
                while(1);

                return 0;

}
```

INPOST.cpp

```
/*
 * inpost.cpp
 *
 *  Created on: 27-Dec-2017
 *      Author: e2002
 */

#include "inpost.h"
#include "stack.h"
#include <string>
#include<math.h>
#include <stdexcept>
#include <iostream>
#include "stack_int.h"
using namespace std;


int inpost ::  priority(char x)
{
	if(x=='^')
		return 4;

	else if(x=='/' || x=='*')
		return 3;

	else if(x=='+' || x=='-')
		return 2;


	else return -1;

}

int inpost :: right_assosciativity(char x)
{
	if(x=='^')
		return 1;

	else
		return 0;
}

void inpost :: infix_to_postfix(string arr, stacks s)
{
	int len=arr.length(),k=0;

		for(int i=0;i<len;i++)
		{
			char x=arr[i];
```

```
        if( (x>='A' && x<='Z') || (x>='a' && x<='z') )
        {
                cout<<""<<x;
        }

        else if(x=='(')
                                s.push(x);

        else if(x==')')
        {
                while(s.tope() != '(')
                {
                        cout<<""<<s.pop();
                }

                s.pop();

        }

        else
        {
                if(s.isempty())
                        s.push(x);

                else
                {
                        if(right_assosciativity(x))
                        {
                                while(priority(s.tope()) > priority(x) )
                                {
                                        cout<<""<<s.pop();

                                }
                        }

                        else
                        {
                                while(priority(s.tope()) >= priority(x) )
                                {
                                        cout<<""<<s.pop();

                                }
                        }

                        s.push(x);
                }
        }
}

while(!s.isempty())
                {
```

```
                                        cout<<""<<s.pop();
                                }

                cout<<endl;
}

int inpost :: postfix_eval(string arr,stack_int s)
{
        int len=arr.length();

        for(int i=0;i<len;i++)
        {
                char x=arr[i];

                if(x>='0' && x<='9')
                        s.push(x-'0');

                else
                {
                        int op2=s.pop();
                        int op1=s.pop();
                        int res=0;

                        switch(x)
                        {
                                case '*':
                                        res=op1*op2;
                                        break;

                                case '+':
                                        res=op1+op2;
                                        break;

                                case '-':
                                        res=op1-op2;
                                        break;

                                case '/':
                                        res=op1/op2;
                                        break;

                                case '^':
                                        res=pow(op1,op2);
                                        break;
                        }

                        s.push(res);
                }
        }

        return s.pop();     }
```

INPOST.h

```
/*
 * inpost.h
 *
 *  Created on: 27-Dec-2017
 *      Author: e2002
 */

#ifndef INPOST_H_
#define INPOST_H_
#include "stack.h"
#include <string>
#include <iostream>
#include <stdexcept>
#include "stack_int.h"
using namespace std;
class inpost
{

public:
		int isoperator(char x);
		int priority(char x);
		int right_assosciativity(char x);

		void infix_to_postfix(string arr,stacks s);
		int postfix_eval(string arr,stack_int p);

};

#endif /* INPOST_H_ */
```

INPRE.h

```
/*
 * inpre.h
 *
 *  Created on: 27-Dec-2017
 *      Author: e2002
 */

#ifndef INPRE_H_
#define INPRE_H_
#include <string>
#include "stack.h"
#include <iostream>
#include <stdexcept>
#include "stack_int.h"
using namespace std;


class inpre
{

public:
        void infix_to_prefix(string arr,stacks s);
        int prefix_eval(string arr,stack_int p);
};

#endif /* INPRE_H_ */
```

INPRE.cpp

```
/*
 * inpre.cpp
 *
 *  Created on: 27-Dec-2017
 *      Author: e2002
 */

#include "inpre.h"
#include "inpost.h"
#include "stack.h"
#include <string>
#include <stdexcept>
#include <iostream>
#include "stack_int.h"
#include<math.h>
using namespace std;

void inpre :: infix_to_prefix(string arr,stacks s)
{
	int len=arr.length(),k=0;

	string ans;

	inpost post;

	for(int i=len-1;i>=0;i--)
	{
	  char x=arr[i];

		if( (x>='A' && x<='Z') || (x>='a' && x<='z') )
		{
			ans[k++]=x;
		}

		else if(x==')')
			s.push(x);

		else if(x=='(')
		{
			while(s.tope() != ')')
			{
				ans[k++]=s.pop();
			}

			s.pop();

		}

		else
```

```
		{
			//cout<<"IGI\n";

			if(s.isempty())
			{
				s.push(x);
			}


			else
			{
				if(post.right_assosciativity(x))
				{
					while(post.priority(s.tope()) >= post.priority(x) )
					{
						ans[k++]=s.pop();

					}
				}

				else
				{
					while(post.priority(s.tope()) > post.priority(x) )
					{
						ans[k++]=s.pop();

					}
				}

				s.push(x);
			}
		}
	}

	while(!s.isempty())
		ans[k++]=s.pop();


	for(int i=k-1;i>=0;i--)
		cout<<""<<ans[i];

	cout<<endl;

}

int inpre :: prefix_eval(string arr,stack_int s)
{
	int len=arr.length();

	for(int i=len-1;i>=0;i--)
	{
		char x=arr[i];
```

```
            if(x>='0' && x<='9')
                s.push(x-'0');

            else
            {
                int op1=s.pop();
                int op2=s.pop();
                int res=0;

                switch(x)
                {
                    case '*':
                        res=op1*op2;
                        break;

                    case '+':
                        res=op1+op2;
                        break;

                    case '-':
                        res=op1-op2;
                        break;

                    case '/':
                        res=op1/op2;
                        break;

                    case '^':
                        res=pow(op1,op2);
                        break;
                }

                s.push(res);
            }
        }

        return s.pop();
}
```

STACKS.h

```
/*
 * stacks.h
 *
 *  Created on: 27-Dec-2017
 *      Author: e2002
 */

#ifndef stacks_H_
#define stacks_H_

#include <stdexcept>
#include <stdexcept>

typedef struct node
{
		char data;
		struct node *link;
}node;

//using node= node<T>

class stacks
{
	int max=50;
	node *Top;

public:
	stacks();

	int isempty();
	int isfull();

	void push(int x);
	char tope();
	char pop();

};

#endif /* stacks_H_ */
```

STACK.cpp

```
/*
 * stacks.cpp
 *
 *  Created on: 27-Dec-2017
 *      Author: e2002
 */

#include "stack.h"
#include<malloc.h>
#include<iostream>
#include <stdexcept>
using namespace std;

stacks::stacks()
{
	Top=NULL;
}

int stacks:: isempty()
{
	if( Top==NULL)
		return 1;

	else
		return 0;
}

int stacks:: isfull()
{
	int count=0;
	node *p;

	p=Top;

	while(p!=NULL)
	{
		count++;
		p=p->link;
	}

	if(count==max)
		return 1;

	else
		return 0;
}

void stacks:: push(int x)
{
	if(!isfull())
```

```
	{
		node *n;

		n=(node*)malloc(sizeof(node));

		n->data=x;
		n->link=Top;

		Top=n;
	}

	else
	{
		cout<<"stacks is Full!!\n";
		return ;
	}
}

char stacks:: tope()
{
	if(!isempty())
		return Top->data;

	else
	{
				return ' ' ;
	}
}

char stacks :: pop()
{
	if(!isempty())
	{
		int n;
		n=Top->data;

		Top=Top->link;

		return n;
	}

	else
		return ' ';
}
```

STACK_INT.h

```
/*
 * stack_int.h
 *
 *  Created on: 01-Jan-2018
 *      Author: e2002
 */

#ifndef STACK_INT_H_
#define STACK_INT_H_

typedef struct node1
{
		int data;
		struct node1 *link;
}node1;

class stack_int
{

	int max=50;
		node1 *Top;

	public:
		stack_int();

		int isempty();
		int isfull();

		void push(int x);
		int tope();
		int pop();
};

#endif /* STACK_INT_H_ */
```

STACK_INT.cpp

```
/*
 * stack_int.cpp
 *
 *  Created on: 01-Jan-2018
 *      Author: e2002
 */

#include "stack_int.h"
#include<malloc.h>
#include<iostream>
#include <stdexcept>
using namespace std;

stack_int::stack_int()
{
	Top=NULL;
}

int stack_int:: isempty()
{
	if( Top==NULL)
		return 1;

	else
		return 0;
}

int stack_int:: isfull()
{
	int count=0;
	node1 *p;

	p=Top;

	while(p!=NULL)
	{
		count++;
		p=p->link;
	}

	if(count==max)
		return 1;

	else
		return 0;
}

void stack_int:: push(int x)
{
```

```
	if(!isfull())
	{
		node1 *n;

		n=(node1*)malloc(sizeof(node1));

		n->data=x;
		n->link=Top;

		Top=n;
	}

	else
	{
		cout<<"Stack is Full!!\n";
		return ;
	}
}

int stack_int:: tope()
{
	if(!isempty())
		return Top->data;

	else
			return -999;
}

int stack_int:: pop()
{
	if(!isempty())
	{
		int n;
		n=Top->data;

		Top=Top->link;

		return n;
	}

	else
				return -999;
}
```