Name: Charu Chandra Joshi Roll No. 202111019

Assignment 2
DS Lab

1. Code

```
#include<iostream>
#include<string.h>
using namespace std;

#define MAX 1000
class Stack{

	int top;

public:
	int a[MAX];

	Stack()
	{
		top = -1;
	}

	bool push(int x){
		if(top >= MAX-1)
		{
			cout << "Stack Overflow\n";
			return false;
		}
		else
		{
```

```
		top++;
		a[top] = x;
		// cout << x << " pushed successfully.\n";
		return true;
	}
}

int pop() {
	if(top < 0)
	{
		cout << "Stack Underflow.\n";
		return 0;
	}
	else
	{
		// cout << a[top] << " will be popped.\n";
		top--;
		return a[top];
	}
}

int peek(){
	if(top < 0)
	{
		cout << "Stack Underflow.\n";
		return 0;
	}
	else
		return a[top];
}

bool isEmpty()
{
	if(top < 0)
		return true;
	else return false;
}
```

```
        void display()
        {
                for(int i = top; i >= 0; --i)
                        cout << a[i] << " ";
                cout << endl;
        }

};

Stack obj;

void ParenthesisBalanced(){
        string s;
        cin >> s;
        bool flag = true;
        cout << "Checking if " << s << " is balanced or not." << endl;
        for(int i = 0; s[i] != '\0'; ++i)
        {

                if(s[i] == '[')
                        obj.push(1);
                else
                        if(s[i] == ']')
                        {
                                if(obj.peek() == 1)
                                        obj.pop();
                                else
                                {
                                        flag = false;
                                        break;
                                }
                        }

                if(s[i] == '{')
                        obj.push(2);
                else
                        if(s[i] == '}')
                        {
```

```
                        if(obj.peek() == 2)
                                obj.pop();
                        else
                        {
                                flag = false;
                                break;
                        }
                }

            if(s[i] == '(')
                obj.push(3);
            else
                if(s[i] == ')')
                {
                        if(obj.peek() == 3)
                                obj.pop();
                        else
                        {
                                flag = false;
                                break;
                        }
                }
        }
        if(!obj.isEmpty())
            flag = false;
        if(flag)
                cout << "Balanced.\n";
        else
            cout << "Not Balanced.\n";
}

int main(){

        ParenthesisBalanced();
        return 0;
}
```

OUTPUT:

```
C:\Users\Charu Chandra Joshi\Desktop\CS102\Lab2>firsti.exe
[{}()]
Checking if [{}()] is balanced or not.
Balanced.
```

1.b.Code

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

int main(){
	string s;
	cin >> s;
	bool flag = false;
	for(int i = 0; i < s.length(); ++i)
	{
		//for []
		if(s[i] == ']')
		{
			s[i] = 'x';
			if(s[i-1] == '[')
			{
				s[i-1] = 'x';
				flag = true;
			}
			else
			if(s[i-1] == 'x')
			{
				int k = i-1;
```

```
                while(s[--k] == 'x')
                        continue;
                if(s[k] == '[')
                {
                        s[k] = 'x';
                        flag = true;
                }
                else flag = false;
        }
        else flag = false;
}

//for {}
if(s[i] == '}')
{
        s[i] = 'x';
        if(s[i-1] == '{')
        {
                s[i-1] = 'x';
                flag = true;
        }
        else
        if(s[i-1] == 'x')
        {
                int k = i-1;
                while(s[--k] == 'x')
                        continue;
                if(s[k] == '{')
                {
                        s[k] = 'x';
                        flag = true;
                }
                else flag = false;
        }
        else flag = false;
}

//for ()
```

```
		if(s[i] == ')')
		{
			s[i] = 'x';
			if(s[i-1] == '(')
			{
				s[i-1] = 'x';
				flag = true;
			}
			else
			if(s[i-1] == 'x')
			{
				int k = i-1;
				while(s[--k] == 'x')
					continue;
				if(s[k] == '(')
				{
					s[k] = 'x';
					flag = true;
				}
				else flag = false;
			}
			else flag = false;
		}
	}
	for(int i = 0; i<s.length(); ++i)
		if(s[i] != 'x')
		{
			flag = false;
			break;
		}
	if(flag)	cout <<"BALANACED";
	else		cout<<"NOT BALANCED";
	return 0;
}
```

OUTPUT:

```
C:\Users\Charu Chandra Joshi\Desktop\CS102\Lab2>firstii.exe
[{}{()}]
BALANACED
C:\Users\Charu Chandra Joshi\Desktop\CS102\Lab2>firstii.exe
[{
NOT BALANCED
C:\Users\Charu Chandra Joshi\Desktop\CS102\Lab2>
```

2. Code

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

#define MAX 1000

class stack{
public:
	int top;
	char a[MAX];

	stack(){ top = -1;}

	bool isEmpty(){
		if(top < 0) return true;
		else return false;
	}

	bool push(char x)
	{
		if(top == MAX-1) return false;
		else
		{
			a[++top] = x;
			return true;
		}
	}
```

```
	char pop(){
		if(top < 0)
		{
			cout << "Stack Underflow";
			return '\0';
		}
		else
			return a[top--];
	}

	char peek(){

		if(top < 0)
		{
			cout <<"Stack Underflow.";
			return '\0';
		}
		else
			return a[top];
	}
}st;

int precedence(char c)
{
	if(c == '^')
		return 3;
	else
		if(c == '/' || c == '*')
			return 2;
		else
			if(c == '+' || c == '-')
				return 1;
}

void postfix(string infix)
{
	string pf;
	char c;
```

```
for(int i = 0; i < infix.length(); ++i)
{
        c = infix[i];
        if((c >= 'a' && c <='z') || (c >= 'A' && c <= 'Z') || (c >= '0' && c <= '9'))
                pf += c;
        else
        if(c == '(')
                st.push('(');
        else
        if(c == ')')
```

```
C:\Users\Charu Chandra Joshi\Desktop\CS102\Lab2>second.exe
A+B-C*D^t
ABC-+Dt^*
```

```
        {
                while(st.peek() != '(')
                {
                        pf += st.peek();
                        st.pop();
                }
                st.pop();
        }

        else{
                while(!st.isEmpty() && precedence(infix[i] <= st.peek()))
                        if(c == '^' && st.peek() == '^')
                                break;
                        else
                {
                        pf += st.peek();
                        st.pop();
                }
                st.push(c);
        }
}
```

```
	while(!st.isEmpty())
		{pf += st.peek();
	st.pop();}

	cout << pf << endl;
}


int main(){

	string infix;
	cin >> infix;
	postfix(infix);
	return 0;
}
```

OUTPUT:

3. Code

```
#include <iostream>
#include <stack>
#include <vector>
using namespace std;
class SpecialStack
{

	// Sentinel value for min
	int min = -1;

	// DEMO_VALUE
	static const int demoVal = 9999;
	stack<int> st;
```

```
public:

        void getMin()
        {
                cout << "min is: " << min << endl;
        }

        void push(int val)
        {

                // If stack is empty OR current element
                // is less than min, update min.
                if (st.empty() || val < min)
                {
                        min = val;
                }

                // Encode the current value with
                // demoVal, combine with min and
                // insert into stack
                st.push(val * demoVal + min);
        }

        int pop()
        {
                // if stack is empty return -1;
                if ( st.empty() ) {
                cout << "stack underflow" << endl ;
                return -1;
                }

                int val = st.top();
                st.pop();
```

```
		// If stack is empty, there would
		// be no min value present, so
		// make min as -1
		if (!st.empty())
			min = st.top() % demoVal;
		else
			min = -1;

		// Decode actual value from
		// encoded value
		return val / demoVal;
	}

	int peek()
	{

		// Decode actual value
		// from encoded value
		return st.top() / demoVal;
	}
};

// Driver Code
int main()
{
	SpecialStack s;
	cout << "Enter the number of elements you want to push :";
	int n;
	cin >> n;

	int count = 0;
	for(int i = 0; i < n; i++)
	{
		int p;
		cin >> p;
```

```
            s.push(p);
            count++;
        }

        cout << endl;
        s.getMin();
        return 0;
}
```

OUTPUT:

```
Enter the number of elements you want to push :5
1 2 3 4 5

min is: 1

C:\Users\Charu Chandra Joshi\Desktop\CS102\Lab2>third.exe
Enter the number of elements you want to push :7
9 8 7 5 64 2 -2

min is: -2
```