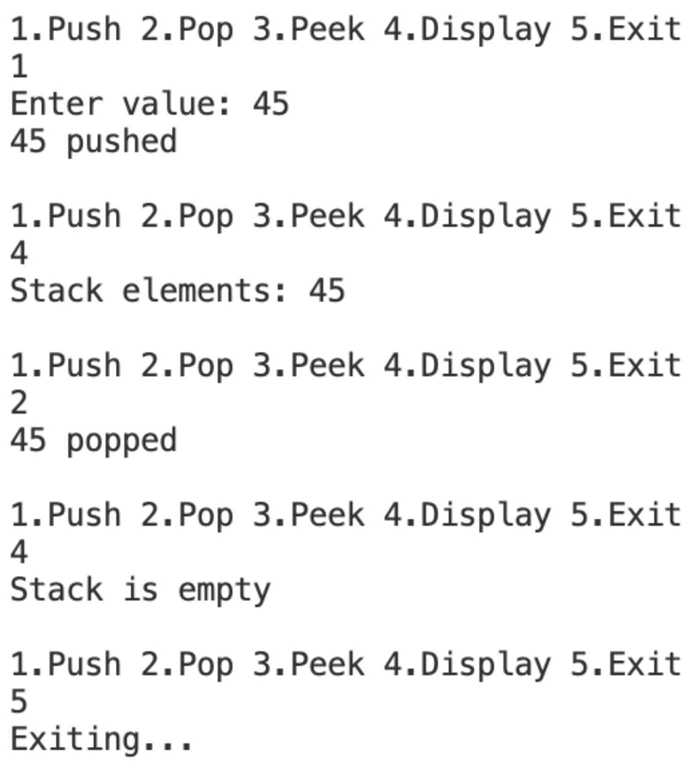
DSA Assignment 3

1.

```cpp
#include <iostream>
using namespace std;
#define MAX 10

int stck[MAX], top = -1;

void push(int x){
    if(top == MAX-1){
        cout<<"Stack is full\n";
    } else {
        stck[++top] = x;
        cout<<x<<" pushed\n";
    }
}

void pop(){
    if(top == -1){
        cout<<"Stack is empty\n";
    } else {
        cout<<stck[top--]<<" popped\n";
    }
}

void peek(){
    if(top == -1){
        cout<<"Stack is empty\n";
    } else {
        cout<<"Top element is "<<stck[top]<<endl;
    }
}

void display(){
    if(top == -1){
        cout<<"Stack is empty\n";
    } else {
        cout<<"Stack elements: ";
        for(int i=0;i<=top;i++){
            cout<<stck[i]<<" ";
        }
        cout<<endl;
    }
}

int main(){
    int choice,val;
    do{
        cout<<"\n1.Push 2.Pop 3.Peek 4.Display 5.Exit\n";
        cin>>choice;
        switch(choice){
            case 1: cout<<"Enter value: "; cin>>val; push(val); break;
            case 2: pop(); break;
            case 3: peek(); break;
            case 4: display(); break;
            case 5: cout<<"Exiting...\n"; break;
            default: cout<<"Wrong choice\n";
        }
    }while(choice!=5);
}
```

OUTPUT

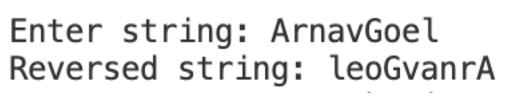


```
1.Push 2.Pop 3.Peek 4.Display 5.Exit
1
Enter value: 45
45 pushed

1.Push 2.Pop 3.Peek 4.Display 5.Exit
4
Stack elements: 45

1.Push 2.Pop 3.Peek 4.Display 5.Exit
2
45 popped

1.Push 2.Pop 3.Peek 4.Display 5.Exit
4
Stack is empty

1.Push 2.Pop 3.Peek 4.Display 5.Exit
5
Exiting...
```

2.

```
#include <iostream>
#include <stack>
using namespace std;
int main(){
    string s;
    cout<<"Enter string: ";
    cin>>s;
    stack<char> st;
    for(char c: s){
        st.push(c);
    }
    cout<<"Reversed string: ";
    while(!st.empty()){
        cout<<st.top();
        st.pop();
    }
    cout<<endl;
}
```

OUTPUT

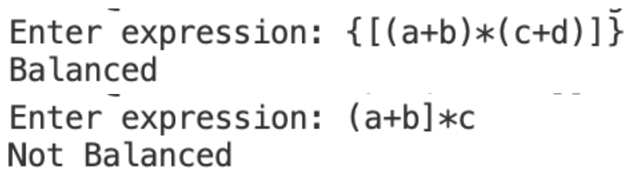


```
Enter string: ArnavGoel
Reversed string: leoGvanrA
```

3.

```
#include <iostream>
#include <stack>
using namespace std;
bool isBalanced(string exp){
    stack<char> st;
    for(char c: exp){
        if(c=='(' || c=='{' || c=='['){
            st.push(c);
        } else if(c==')' || c=='}' || c==']'){
            if(st.empty()) return false;
            char top = st.top();
            if((c==')' && top=='(') || (c=='}' && top=='{') || (c==']' &&
top=='[')){
                st.pop();
            } else return false;
        }
    }
    return st.empty();
}
int main(){
    string exp;
    cout<<"Enter expression: ";
    cin>>exp;
    if(isBalanced(exp)) cout<<"Balanced\n";
    else cout<<"Not Balanced\n";
}
```

OUTPUT



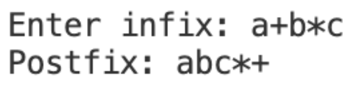
```
Enter expression: {[(a+b)*(c+d)]}
Balanced
Enter expression: (a+b]*c
Not Balanced
```

4.

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

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

string infixToPostfix(string s){
    stack<char> st;
    string res;
    for(char c: s){
        if((c>='a' && c<='z') || (c>='A' && c<='Z') || (c>='0' && c<='9')){
            res+=c;
        }
        else if(c=='('){
            st.push(c);
        }
        else if(c==')'){
```

```cpp
            while(!st.empty() && st.top()!='('){
                res+=st.top();
                st.pop();
            }
            st.pop();
        }
        else{
            while(!st.empty() && prec(st.top())>=prec(c)){
                res+=st.top();
                st.pop();
            }
            st.push(c);
        }
    }
    while(!st.empty()){
        res+=st.top();
        st.pop();
    }
    return res;
}

int main(){
    string exp;
    cout<<"Enter infix: ";
    cin>>exp;
    cout<<"Postfix: "<<infixToPostfix(exp)<<endl;
}
```

OUTPUT



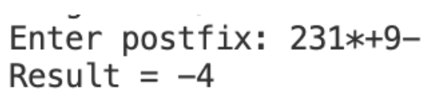
```
Enter infix: a+b*c
Postfix: abc*+
```

5.

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

int evaluatePostfix(string exp){
    stack<int> st;
    for(char c: exp){
        if(isdigit(c)){
            st.push(c-'0'); // convert char to int
        }
        else{
            int val2=st.top(); st.pop();
            int val1=st.top(); st.pop();
            switch(c){
                case '+': st.push(val1+val2); break;
                case '-': st.push(val1-val2); break;
                case '*': st.push(val1*val2); break;
                case '/': st.push(val1/val2); break;
            }
        }
    }
    return st.top();
}
```

```cpp
int main(){
    string exp;
    cout<<"Enter postfix: ";
    cin>>exp;
    cout<<"Result = "<<evaluatePostfix(exp)<<endl;
}
```

OUTPUT



```
Enter postfix: 231*+9-
Result = -4
```