Data Structures (UCS301) Assignment

NAME-Suneet Arora ROLL NO-1024030174

```
1. #include<iostream>


using namespace std;

int top = -1;
const int n = 10;

int stack[n];
void push(int x){

    top++;
    stack[top] = x;

}
void pop(){
    top--;
}
bool isEmpty(){
    if(top==-1)
    {
        return true;
    }
    else{
        return false;
    }
}
bool isFull(){
    if(top==10)
    {
        return true;
    }
    else{
        return false;
    }
}
void display(){
    for(int i=top;i>=0;i--){
        cout<<stack[i]<<endl;
    }
}
void peek(){
```

```
    cout<<stack[top];
}
int main(){
```

```
    int ch;
    while(ch!=0)
    {
        cout<<endl<<endl;
    cout<<"Enter 1 to push"<<endl;
           cout<<"Enter 2 to pop"<<endl;
          cout<<"Enter 3 to check if empty"<<endl;
          cout<<"Enter 4 to check if full"<<endl;
          cout<<"Enter 5 to display"<<endl;
          cout<<"Enter 6 to peek"<<endl;
          cout<<"Enter 0 to exit"<<endl;
           cin>>ch;
    cout<<endl<<endl;


    switch (ch)
    {
    case 1:
        int p;
        cout<<"Enter element to push:";
        cin>>p;
        push(p);
        break;
    case 2:
    pop();
    break;

    case 3:
    cout<<isEmpty();
    break;
    case 4:
    cout<<isFull();
    break;
    case 5:
    display();
    break;
    case 6:
    peek();
    break;


    }

    }
}
```

2.
```cpp
#include <iostream>

using namespace std;

int top = -1;
void push(char stack[], int x)
{
    top++;
    stack[top] = x;
}
void pop()
{
    top--;
}

int main()
{

    string s;
    cout << "Enter the string: ";
    getline(cin, s);
    const int n = s.length();
    char stack[n];

    for (int i = 0; i < n; i++)
    {
        push(stack, s[i]);
    }
    int h = 0;
    for (int i = top; i >= 0; i--)
    {
        s[h] = stack[top];

        h++;
        pop();
    }

    for (int i = 0; i < n; i++)
    {
        cout << s[i];
    }

    return 0;
}
```

3.
```cpp
#include <iostream>

#include <stack>
using namespace std;

bool isBalanced(string expr) {
    stack<char> st;

    for(char c : expr) {

        if(c == '(' || c == '{' || c == '[') {
            st.push(c);
        }

        else if(c == ')' || c == '}' || c == ']') {
            if(st.empty()) return false;

            char top = st.top();
            st.pop();

            if((c == ')' && top != '(') ||
               (c == '}' && top != '{') ||
               (c == ']' && top != '[')) {
                return false;
            }
        }
    }

    return st.empty();
}

int main() {
    string expr;
    cout << "Enter expression: ";
    cin >> expr;

    if(isBalanced(expr))
        cout << "Balanced" << endl;
    else
        cout << "Not Balanced" << endl;

    return 0;
}
```

```
4.#include <iostream>

#include <stack>
using namespace std;


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

int main() {
    string infix, postfix = "";
    cout << "Enter infix expression: ";
    cin >> infix;

    stack<char> st;

    for(char c : infix) {

                                        if((c >= 'a' && c <= 'z') || (c >= 'A' &&
                                        c <= 'Z') || (c >= '0' && c <=
'9')) {

                                            postfix += c;
                                        }


        else if(c == '(') {
            st.push(c);
        }

        else if(c == ')') {
            while(!st.empty() && st.top() != '(') {
                postfix += st.top();
                st.pop();
            }
            st.pop();
        }

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

```
        }
    }


    while(!st.empty()) {
        postfix += st.top();
        st.pop();
    }

cout << "Postfix expression: " << postfix << endl; return
    0;
}
```

```
5. #include <iostream>

#include <stack>
using namespace std;

int evaluatePostfix(string exp) {
    stack<int> st;

for(char c : exp) {
// If operand (digit) if(isdigit(c)) {
st.push(c - '0');  // convert char to int
        }
else {
// Pop top 2 elements
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();
}
```

```
int main() {
    string exp;
    cout << "Enter postfix expression: ";
    cin >> exp;

    cout << "Result = " << evaluatePostfix(exp) << endl;
    return 0;
}
```