DSA Assignment 3

Name: Jashnoor Singh
Roll No: 1024030106

Answer 1:

```
#include <iostream>
using namespace std;
class Stack{
    public:
    int arr[5];
    int top;
    Stack(){
        top=-1;
    }
    void push(int n){
        if(isFull()){
            cout<<"Stack is full cannot push anything ";
        }
        else{
            arr[++top]=n; // first increases the value of top then inserts the value
            cout<<"The element is pushed";
        }
    }
    void pop(){
        if(isEmpty()){
            cout<<"There is nothing in array to delete ";
        }
        else{
            cout<<"The element "<<arr[top--]<<" is removed from the array";
        }
    }
    bool isEmpty(){
        if(top==-1){
            return true;
        }
        else
        return false;
    }
    bool isFull(){
        if(top==4){
            return true;
        }
        else
        return false;
    }
    void display(){
        if(isEmpty()){
            cout<<"There is nothing to display array is empty";
        }
        else{
            for(int i=0;i<5;i++){
                cout<<arr[i]<<' ';
            }
        }
```

```
    }
    void peek(){
        if(isEmpty()){
            cout<<"There is nothing to peek array is empty";
        }
        else{
            cout<<"The top element is "<<arr[4];
        }
    }
};
int main(){
    Stack s;
    int choice,value;
    cout<<"To push element into stack press 1\nTo pop element from stack press 2\nTo check if stack is empty press 3\nTo check if stack is full press 4\nTo display elements in array press 5\nTo peek top element press 6\nTo exit program press 7";
    while(choice!=7){
        cout<<"\nEnter your choice: ";
        cin>>choice;
    switch(choice){
        case 1:
        cout<<"Enter value you want to insert"<<endl;
        cin>>value;
        s.push(value);
        break;

        case 2:
        s.pop();
        break;

        case 3:
        if(s.isEmpty()){
            cout<<"Stack is empty";
        }
        else{
            cout<<"Stack is not empty";
        }
        break;

        case 4:
        if(s.isFull()){
            cout<<"Stack is full ";
        }
        else{
            cout<<"Stack is not full";
        }
        break;

        case 5:
        s.display();
        break;

        case 6:
        s.peek();
        break;

        case 7:
        break;

        default:
```

```
        cout<<"Enter a valid value";
    }
    cout<<"\nIf you wish to exit the program press 7";
}

}
```

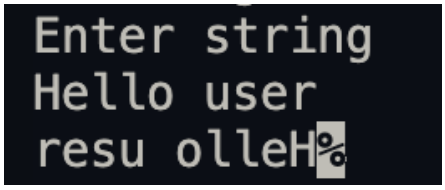
Output

```
To push element into stack press 1
To pop element from stack press 2
To check if stack is empty press 3
To check if stack is full press 4
To display elements in array press 5
To peek top element press 6
To exit program press 7
Enter your choice: 1
Enter value you want to insert
2
The element is pushed
If you wish to exit the program press 7
Enter your choice: 1
Enter value you want to insert
3
The element is pushed
If you wish to exit the program press 7
Enter your choice: 1
Enter value you want to insert
4
The element is pushed
If you wish to exit the program press 7
Enter your choice: 1
Enter value you want to insert
5
The element is pushed
If you wish to exit the program press 7
Enter your choice: 1
Enter value you want to insert
6
The element is pushed
If you wish to exit the program press 7
Enter your choice: 1
Enter value you want to insert
7
Stack is full cannot push anything
If you wish to exit the program press 7
Enter your choice: 2
The element 6 is removed from the array
If you wish to exit the program press 7
Enter your choice: 3
Stack is not empty
If you wish to exit the program press 7
Enter your choice: 4
Stack is not full
If you wish to exit the program press 7
Enter your choice: 5
2 3 4 5 6
If you wish to exit the program press 7
Enter your choice: 6
The top element is 6
If you wish to exit the program press 7
Enter your choice: 7

If you wish to exit the program press 7%
```

Answer 2

```
// Reverse string using a stack
#include <iostream>
using namespace std;
int main(){
    string s;
    string p;
    cout<<"Enter string\n";
    getline(cin,s);
    int a=s.length();
    int i=0;
    while(a!=0){
        p.push_back(s.back()); // this function uses push back to increase space in p which was initially zero and equates previous space with s.back() ie last element of s
        s.pop_back(); // pop_back is a void function and doesnot have a return type so we cant equate it directly therefore first we use back function to get last character of s and equate it to first character of p and then use pop_back
        a--;
    }
    while(i!=p.length()){
        cout<<p[i];
        i++;
    }
}
```

Output

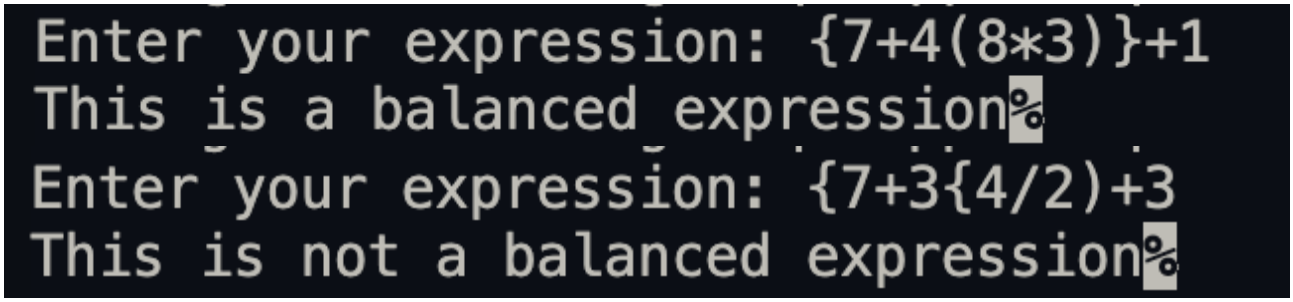


Answer 3

```
// Write a program to check if expression has balanced parenthesis
#include <iostream>
using namespace std;
int main(){
    string s;
    int count=0;
    cout<<"Enter your expression: ";
    getline(cin,s);
    int i=0;
    while(i!=s.length()){
        if(s[i]=='('){
            count++;
            i++;
        }
        else if(s[i]=='{'){
            count++;
            i++;
        }
        else if(s[i]=='['){
            count++;
            i++;
```

```
        }
        else if(s[i]==']'){
            count--;
            i++;
        }
        else if(s[i]=='}'){
            count--;
            i++;
        }
        else if(s[i]==')'){
            count--;
            i++;
        }
        else{
            i++;
            continue;
        }
    }
    if (count!=0){
        cout<<"This is not a balanced expression";
    }
    else{
        cout<<"This is a balanced expression";
    }
}
```

Output

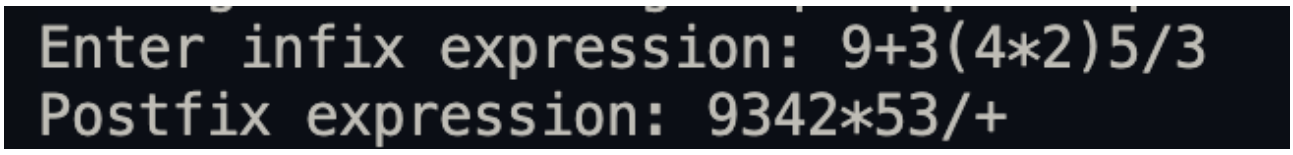


Answer 4

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

int main() {
    string infix;
    cout<<"Enter infix expression: ";
    cin>>infix;
    stack<char> st;
    string postfix="";
    for(char c:infix){
        if (isalnum(c)){
            postfix+=c;
        }
```

```
        else if(c=='('){
            st.push(c);
        }
        else if(c==')'){
            while (!st.empty() && st.top()!='(') {
                postfix+=st.top();
                st.pop();
            }
            if (!st.empty()) st.pop();
        }
        else{
            int prec;
            if (c=='+' || c=='-') prec = 1;
            else if (c=='*' || c=='/') prec = 2;
            else if (c=='^') prec = 3;
            else prec=0;
            while (!st.empty()){
                int topPrec;
                if (st.top()=='+' || st.top()=='-') topPrec = 1;
                else if (st.top()=='*' || st.top()=='/') topPrec = 2;
                else if (st.top()=='^') topPrec = 3;
                else topPrec=0;

                if ((prec<topPrec) || (prec==topPrec && c!='^')) {
                    postfix+=st.top();
                    st.pop();
                } else break;
            }
            st.push(c);
        }
    }

    while (!st.empty()){
        postfix+=st.top();
        st.pop();
    }
    cout << "Postfix expression: " << postfix << endl;
}
```

Output



Answer 5

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

int main() {
    string postfix;
```

```
    cout<<"Enter postfix expression: ";
    cin>>postfix;
    stack<int> st;
    for (char c:postfix){
        if (isdigit(c)){
            st.push(c - '0');
        }
        else{
            int val2=st.top();
            st.pop();
            int val1=st.top();
            st.pop();
            int result;
            switch(c){
                case '+': result = val1 + val2; break;
                case '-': result = val1 - val2; break;
                case '*': result = val1 * val2; break;
                case '/': result = val1 / val2; break;
                case '^': result = pow(val1, val2); break;
                default: cout << "Invalid operator"; return 1;
            }
            st.push(result);
        }
    }
    cout<<"Result = "<<st.top()<<endl;
}
```

Output