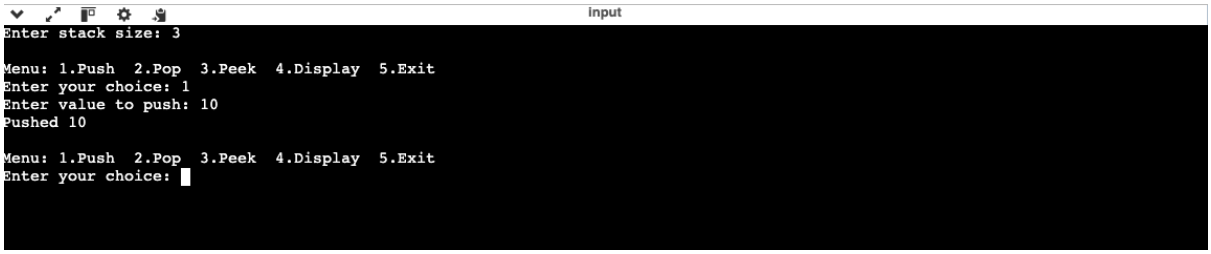
Ques 1

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

class Stack {
   int *stack;
   int top;
   int size;

public:
   Stack(int s) {
      size = s;
      stack = new int[size];
      top = -1;
   }

   bool isEmpty() {
      return top == -1;
   }

   bool isFull() {
      return top == size - 1;
   }

   void push(int value) {
      if (isFull())
         cout << "Stack Overflow!" << endl;
      else {
         stack[++top] = value;
         cout << "Pushed " << value << endl;
      }
   }

   void pop() {
      if (isEmpty())
         cout << "Stack Underflow!" << endl;
      else
         cout << "Popped " << stack[top--] << endl;
   }

   void peek() {
      if (isEmpty())
         cout << "Stack is Empty!" << endl;
      else
         cout << "Top Element: " << stack[top] << endl;
   }
```

```
    void display() {
        if (isEmpty())
            cout << "Stack is Empty!" << endl;
        else {
            cout << "Stack elements: ";
            for (int i = 0; i <= top; i++)
                cout << stack[i] << " ";
            cout << endl;
        }
    }

    ~Stack() {
        delete[] stack;
    }
};

int main() {
    int size;
    cout << "Enter stack size: ";
    cin >> size;

    Stack s(size);
    int choice, val;

    while (true) {
        cout << "\nMenu: 1.Push  2.Pop  3.Peek  4.Display  5.Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
        case 1:
            cout << "Enter value to push: ";
            cin >> val;
            s.push(val);
            break;
        case 2:
            s.pop();
            break;
        case 3:
            s.peek();
            break;
        case 4:
            s.display();
            break;
        case 5:
            cout << "Exiting program..." << endl;
```

```
        return 0;
      default:
        cout << "Invalid choice!" << endl;
      }
   }
}
```

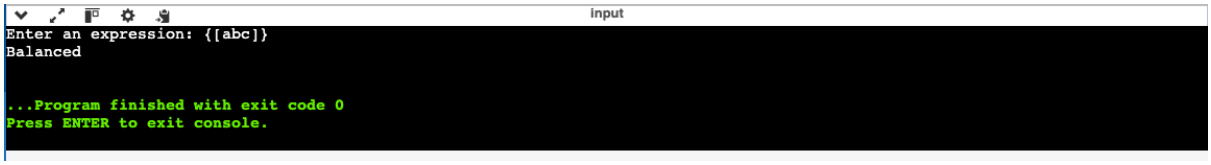


ques 2

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

string reverseString(const string &s) {
   stack<char> st;


   for (char ch : s)
      st.push(ch);

   string reversed = "";


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

   return reversed;
}

int main() {
   string str;
   cout << "Enter a string: ";
   getline(cin, str);  // allows spaces too

   cout << "Reversed String: " << reverseString(str) << endl;
   return 0;
}
```

ques3

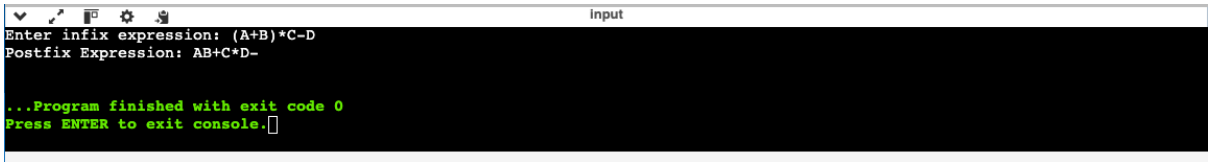
```
#include <iostream>
#include <stack>
#include <string>
using namespace std;

bool isBalanced(const string &exp) {
   stack<char> s;
   for (char ch : exp) {
      if (ch == '(' || ch == '{' || ch == '[') {
         s.push(ch);
      }
      else if (ch == ')' || ch == '}' || ch == ']') {
         if (s.empty())
            return false;

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

         if ((ch == ')' && top != '(') ||
            (ch == '}' && top != '{') ||
            (ch == ']' && top != '['))
            return false;
      }
   }
   return s.empty();
}

int main() {
   string exp;
   cout << "Enter an expression: ";
   getline(cin, exp);

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

   return 0;
}
```



Ques 4

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

int precedence(char op) {
   if (op == '+' || op == '-') return 1;
   if (op == '*' || op == '/') return 2;
   if (op == '^') return 3;
   return 0;
}

string infixToPostfix(const string &exp) {
   stack<char> s;
   string result = "";

   for (char ch : exp) {
      if (isalnum(ch)) {
         result += ch;
      }
      else if (ch == '(') {
         s.push(ch);
      }
      else if (ch == ')') {
         while (!s.empty() && s.top() != '(') {
            result += s.top();
            s.pop();
         }
         if (!s.empty()) s.pop();
      }
      else {
         while (!s.empty() && precedence(s.top()) >= precedence(ch)) {
            result += s.top();
            s.pop();
         }
         s.push(ch);
      }
   }
```

```
    while (!s.empty()) {
        result += s.top();
        s.pop();
    }

    return result;
}

int main() {
    string exp;
    cout << "Enter infix expression: ";
    getline(cin, exp);

    cout << "Postfix Expression: " << infixToPostfix(exp) << endl;
    return 0;
}
```



Ques 5

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

int evaluatePostfix(const string &exp) {
    stack<int> s;

    for (char ch : exp) {
        if (isdigit(ch)) {
            s.push(ch - '0');
        }
        else {
            int b = s.top(); s.pop();
            int a = s.top(); s.pop();

            switch (ch) {
                case '+': s.push(a + b); break;
                case '-': s.push(a - b); break;
                case '*': s.push(a * b); break;
```

```
            case '/': s.push(a / b); break;
            default:
               cout << "Invalid operator: " << ch << endl;
               return -1;
         }
      }
   }
   return s.top();
}

int main() {
   string exp;
   cout << "Enter postfix expression (single-digit numbers): ";
   getline(cin, exp);

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