

Name – Shubham

UID – 22BCS15490

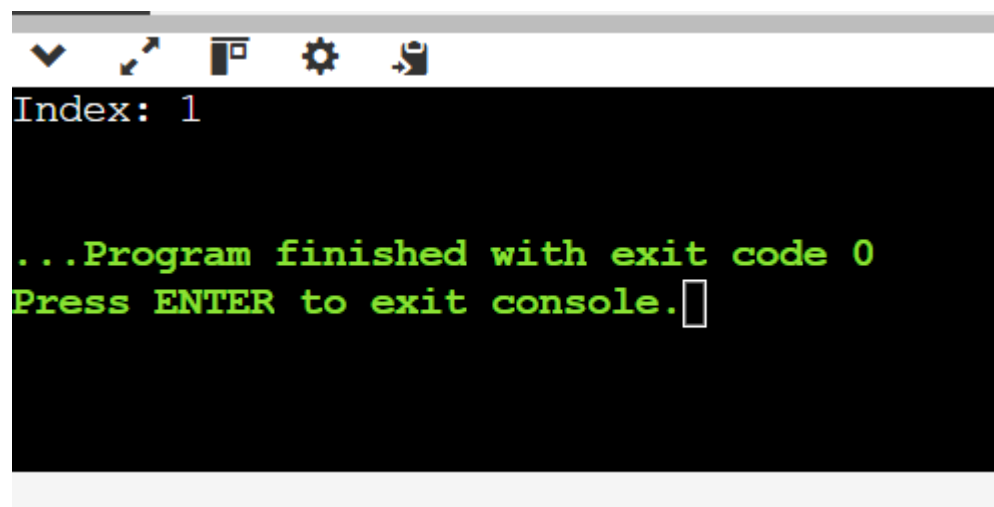
Section – 620-B

Q1. Given a string find the first non repeating character and return their index value if it does not exist than return -1.

Ans

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;
int firstNonRepeatingCharacter(const string& s) {
    vector<int> charCount(256, 0);
    for (char c : s) {
        charCount[c]++;
    }
    for (int i = 0; i < s.length(); ++i) {
        if (charCount[s[i]] == 1) {
            return i;
        }
    }
    return -1;
}
int main() {
    string s = "avacado";
    int index = firstNonRepeatingCharacter(s);
    cout << "Index: " << (index != -1 ? to_string(index) : "No non-repeating character") << endl;
    return 0;
}
```

Output

A screenshot of a console window with a dark background. The text is displayed in a monospaced font. The first line is 'Index: 1' in white. The second line is '...Program finished with exit code 0' in green. The third line is 'Press ENTER to exit console.' in green, followed by a white cursor box.

```
Index: 1

...Program finished with exit code 0
Press ENTER to exit console.
```

Q2.Implementation of 2 queue

Ans

```
#include <iostream>
#include <queue>
using namespace std;
class StackUsingQueues {
private:
    queue<int> queue1, queue2;

public:
    void push(int x) {
        queue1.push(x);
    }

    int pop() {
        while (queue1.size() > 1) {
            queue2.push(queue1.front());
```

```

        queue1.pop();
    }
    int popped_element = queue1.front();
    queue1.pop();
    swap(queue1, queue2);
    return popped_element;
}

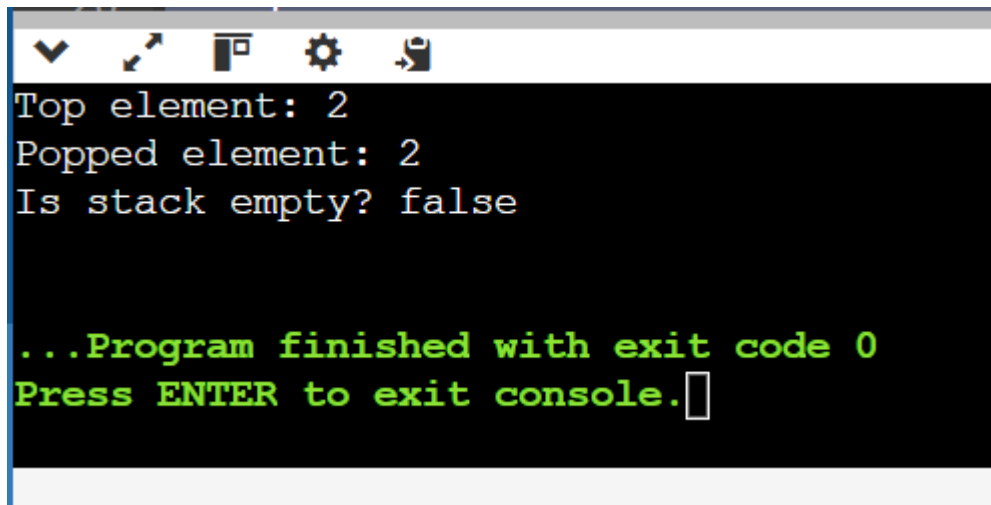
int top() {
    return queue1.back();
}

bool empty() {
    return queue1.empty();
}
};

int main() {
    StackUsingQueues stack;
    stack.push(1);
    stack.push(2);
    cout << "Top element: " << stack.top() << std::endl;
    cout << "Popped element: " << stack.pop() << std::endl;
    cout << "Is stack empty? " << std::boolalpha << stack.empty() << std::endl;
    //false
    return 0;
}

```

## Output

A screenshot of a console window with a dark background and light-colored text. The window has a title bar with standard OS icons. The output text is as follows:

```
Top element: 2
Popped element: 2
Is stack empty? false

...Program finished with exit code 0
Press ENTER to exit console.
```

Q3. Reversal of a string

Ans

```
#include <iostream>

#include <stack>

#include <string>

using namespace std;

string reverseStringUsingStack(const string& input) {

    stack<char> charStack;

    for (char ch : input) {

        charStack.push(ch);

    }

    string reversed;

    while (!charStack.empty()) {

        reversed += charStack.top();

        charStack.pop();

    }

    return reversed;

}
```

```

int main() {
    string input = "Hello";
    string reversed = reverseStringUsingStack(input);

    cout << "Original string: " << input << endl;
    cout << "Reversed string: " << reversed << endl;

    return 0;
}

```

Output

```

Original string: Hello
Reversed string: olleH

...Program finished with exit code 0
Press ENTER to exit console.

```

Q4. Implementation of stack using array and linked list

Ans

```

#include <iostream>
using namespace std;

```

```

class Stack {
private:
    int top;
    int arr[1000];

```

public:

```
Stack() { top = -1; }
```

```
void push(int x) {
```

```
    if (top >= 999) {
```

```
        cout << "Stack Overflow" << endl;
```

```
        return;
```

```
    }
```

```
    arr[++top] = x;
```

```
}
```

```
int pop() {
```

```
    if (top < 0) {
```

```
        cout << "Stack Underflow" << endl;
```

```
        return -1;
```

```
    }
```

```
    return arr[top--];
```

```
}
```

```
int peek() {
```

```
    if (top < 0) {
```

```
        cout << "Stack is Empty" << endl;
```

```
        return -1;
```

```
    }
```

```
    return arr[top];
```

```
}
```

```
bool isEmpty() {
```

```
    return (top < 0);
```

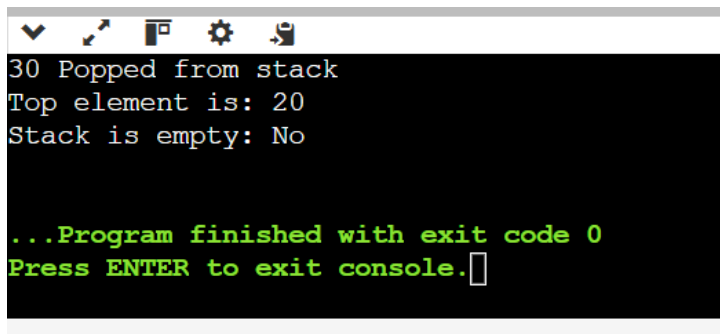
```

    }
};

int main() {
    Stack s;
    s.push(10);
    s.push(20);
    s.push(30);
    cout << s.pop() << " Popped from stack\n";
    cout << "Top element is: " << s.peek() << endl;
    cout << "Stack is empty: " << (s.isEmpty() ? "Yes" : "No") << endl;
    return 0;
}

```

Output



```

30 Popped from stack
Top element is: 20
Stack is empty: No

...Program finished with exit code 0
Press ENTER to exit console.

```

Q5. Implementation of stack by using array only push operation

Ans

```

#include <iostream>
using namespace std;
class Stack {
private:
    int* arr;
    int capacity;

```

```

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

    void push(int value) {
        if (top >= capacity - 1) {
            cout << "Stack overflow! Cannot push " << value << endl;
        } else {
            arr[++top] = value;
            cout << value << " pushed to stack." << endl;
        }
    }
};

int main() {
    int stackSize;
    cout<<"Enter the stack size:"<<endl;
    cin>>stackSize;
    Stack myStack(stackSize);
    int arr[stackSize];
    cout<<"Enter the elements of stack:"<<endl;
    for(int i=0;i<stackSize;i++)
    {
        cin>>arr[i];
    }
    for(int i=0;i<stackSize;i++)
    {

```

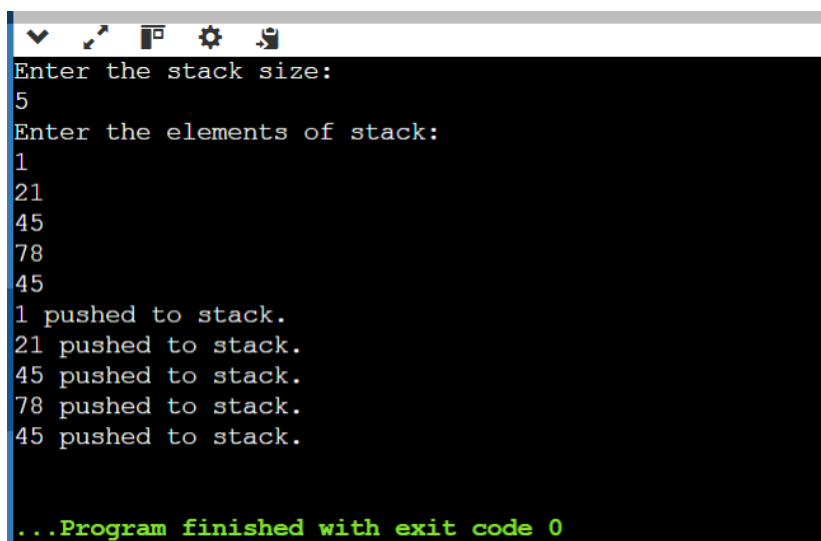


```

        myStack.push(arr[i]);
    }
    return 0;
}

```

Output



```

Enter the stack size:
5
Enter the elements of stack:
1
21
45
78
45
1 pushed to stack.
21 pushed to stack.
45 pushed to stack.
78 pushed to stack.
45 pushed to stack.

...Program finished with exit code 0

```

Q6. The school cafeteria offers circular and square sandwiches at lunch break, referred to by numbers 0 and 1 respectively. All students stand in a queue. Each student either prefers square or circular sandwiches.

Ans

```
#include <iostream>
```

```
#include <queue>
```

```
#include <vector>
```

```
using namespace std;
```

```
int countStudents(vector<int>& students, vector<int>& sandwiches) {
```

```
    queue<int> studentQueue;
```

```
    for (int student : students) {
```

```
        studentQueue.push(student);
```

```
}
```

```
int sandwichIndex = 0;
```

```
int attempts = 0;
```

```
while (!studentQueue.empty() && attempts < studentQueue.size()) {
```

```
    if (studentQueue.front() == sandwiches[sandwichIndex]) {
```

```
        studentQueue.pop();
```

```
        sandwichIndex++;
```

```
        attempts = 0;
```

```
    } else {
```

```
        studentQueue.push(studentQueue.front());
```

```
        studentQueue.pop();
```

```
        attempts++;
```

```
    }
```

```
}
```

```
return studentQueue.size();
```

```
}
```

```
int main() {
```

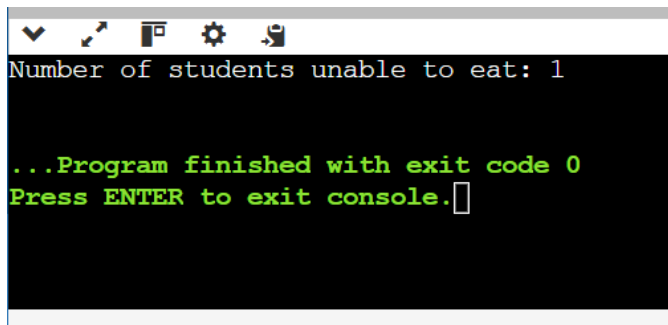
```
    vector<int> students = {1, 1, 0, 0};
```

```
    vector<int> sandwiches = {0, 1, 1, 1};
```

```
    cout << "Number of students unable to eat: " << countStudents(students, sandwiches) << endl;
```

```
    return 0;
```

```
} Output
```

A screenshot of a terminal window with a dark background. The text is displayed in a monospaced font. The first line is "Number of students unable to eat: 1" in white. The second line is "...Program finished with exit code 0" in green. The third line is "Press ENTER to exit console." in green, followed by a cursor icon (a small white rectangle).

Q7. Check the minimum value in stack.

Value are {18,19,29,16,15} output {18}

Ans

```
#include <iostream>
```

```
#include <stack>
```

```
using namespace std;
```

```
int main() {
```

```
    stack<int> s;
```

```
    s.push(18);
```

```
    s.push(19);
```

```
    s.push(29);
```

```
    s.push(16);
```

```
    s.push(15);
```

```
    stack<int> tempStack;
```

```
    int minVal=18;
```

```
    while (!s.empty()) {
```

```
        int x=s.top();
```

```
        s.pop();
```

```
        if(x<=minVal){
```

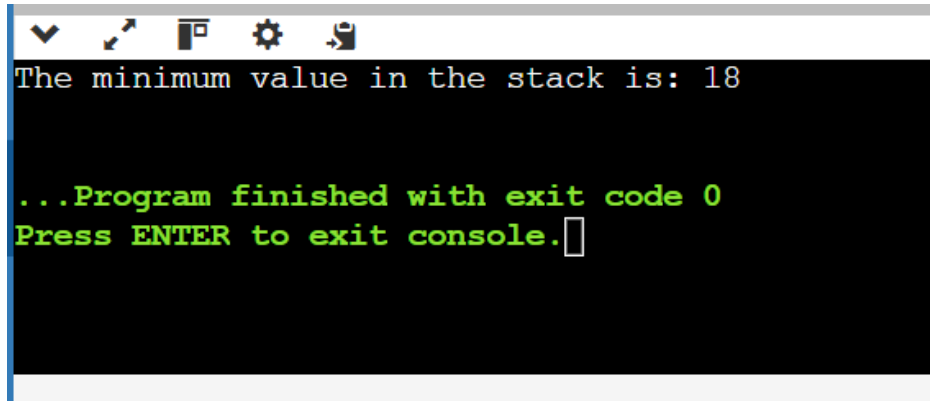
```
            x=minVal;
```

```
        }
```

```
    }
```

```
    cout << "The minimum value in the stack is: " << minVal << endl;
    return 0;
}
```

Output



```
The minimum value in the stack is: 18

...Program finished with exit code 0
Press ENTER to exit console.
```

Q8. Given a queue, write a recursive function to reverse it.

Standard operations allowed :

enqueue(x) : Add an item x to rear of queue.

dequeue() : Remove an item from front of queue.

empty() : Checks if a queue is empty or not.

Ans

```
#include <iostream>
```

```
#include <queue>
```

```
using namespace std;
```

```
void reverseQueue(queue<int>& q) {
```

```
    if (q.empty()) {
```

```
        return;
```

```
    }
```

```
    int front = q.front();
```

```
    q.pop();
```

```
    reverseQueue(q);
```

```
    q.push(front);
```

```
}
```

```

int main() {
    queue<int> q;
    q.push(5);
    q.push(24);
    q.push(9);
    q.push(6);
    q.push(8);
    q.push(4);
    q.push(1);
    q.push(8);
    q.push(3);
    q.push(6);
    reverseQueue(q);
    while (!q.empty()) {
        cout << q.front() << " ";
        q.pop();
    }

    return 0;
}

```

Output

```

6 3 8 1 4 8 6 9 24 5
...Program finished with exit code 0
Press ENTER to exit console.

```

Q9. Given a balanced parentheses string *s*, return the score of the string.

Ans

```
#include <iostream>

#include <stack>

#include <string>

using namespace std;

int scoreOfParentheses(string s) {

    stack<int> st;

    st.push(0);

    for (char c : s) {

        if (c == '(') {

            st.push(0);

        } else {

            int v = st.top();

            st.pop();

            int w = st.top();

            st.pop();

            st.push(w + max(2 * v, 1));

        }

    }

    return st.top();

}

int main() {

    string s1 = "()";

    string s2 = "(() )";

    string s3 = "()( )";
```

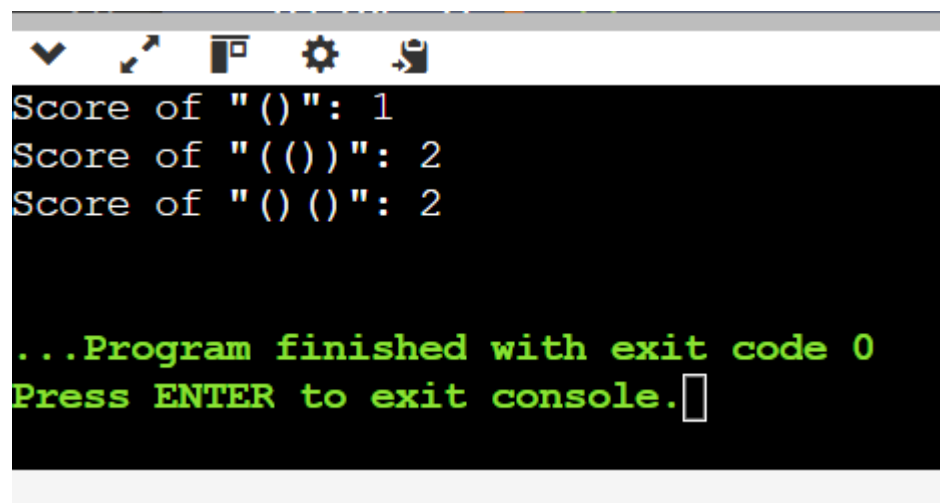
```

cout << "Score of \"\" << s1 << "\": " << scoreOfParentheses(s1) << endl;
cout << "Score of \"\" << s2 << "\": " << scoreOfParentheses(s2) << endl;
cout << "Score of \"\" << s3 << "\": " << scoreOfParentheses(s3) << endl;

return 0;
}

```

Output



```

Score of "()" : 1
Score of "()" : 2
Score of "()" : 2

...Program finished with exit code 0
Press ENTER to exit console.

```

Q10. Given a string containing just the characters '(' and ')', return the length of the longest valid (well-formed) parentheses substring.

Ans

```

#include <iostream>

#include <stack>

#include <string>

using namespace std;

int longestValidParentheses(string s) {
    stack<int> st;

    st.push(-1); // Initial base for calculating valid lengths

    int maxLength = 0;

```

```

for (int i = 0; i < s.length(); ++i) {
    if (s[i] == '(') {
        st.push(i);
    } else {
        st.pop();
        if (st.empty()) {
            st.push(i);
        } else {
            maxLength = max(maxLength, i - st.top());
        }
    }
}

return maxLength;
}

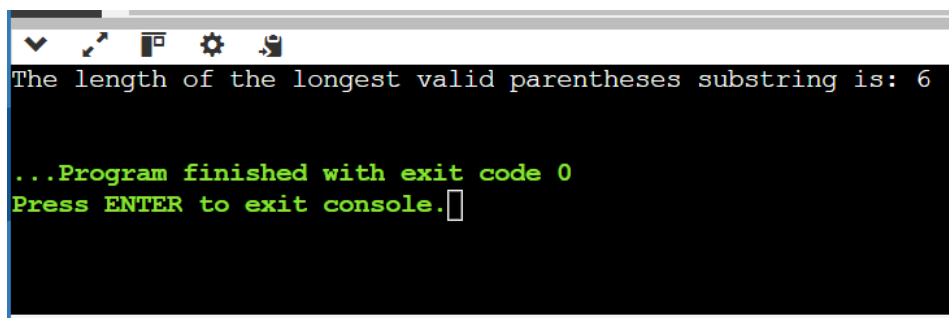
int main() {
    string s = "(()";

    cout << "The length of the longest valid parentheses substring is: " <<
    longestValidParentheses(s) << endl; // Output: 2

    return 0;
}

```

Output



```

The length of the longest valid parentheses substring is: 6

...Program finished with exit code 0
Press ENTER to exit console.

```