

Day - 4

Name – Ishita

Faculty Name- Er. Rajni Devi

UID- 22BCS15353

Date- 24 Dec ,2024

Section – 620-B

Ques 1:

Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

Code:

```
#include <iostream>
#include <stack>
using namespace std;
class MinStack {
private:
    stack<int> s;    // Main stack to hold values
    stack<int> minStack; // Stack to hold the minimum values
public:
    MinStack() {
        // Constructor does not need to initialize anything specific
    }
    void push(int val) {
        s.push(val);
        if (minStack.empty() || val <= minStack.top()) {
            minStack.push(val);
        }
    }
    void pop() {
        if (s.top() == minStack.top()) {
            minStack.pop();
        }
        s.pop();
    }
}
```

```

int top() {
    return s.top();
}

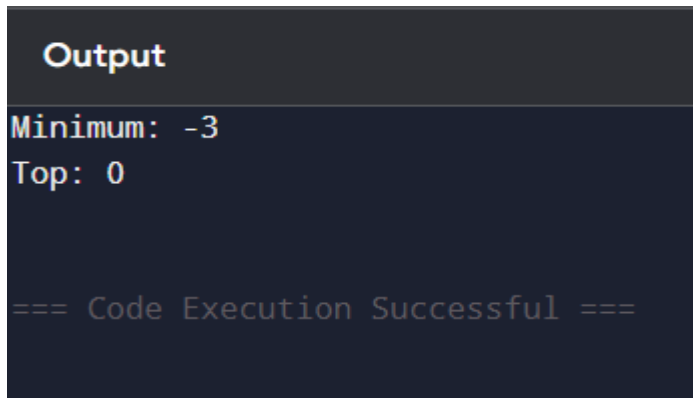
int getMin() {
    return minStack.top();
}

};

int main() {
    MinStack minStack;
    minStack.push(2);
    minStack.push(0);
    minStack.push(-3);
    cout << "Minimum: " << minStack.getMin() << endl; // Should print -3
    minStack.pop();
    cout << "Top: " << minStack.top() << endl;    // Should print 0
    return 0;
}

```

Output:



The screenshot shows a dark-themed window titled "Output". It contains the following text:

```

Minimum: -3
Top: 0

=== Code Execution Successful ===

```

Ques 2:

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.

The number of sandwiches in the cafeteria is equal to the number of students. The sandwiches are placed in a stack.

Code:

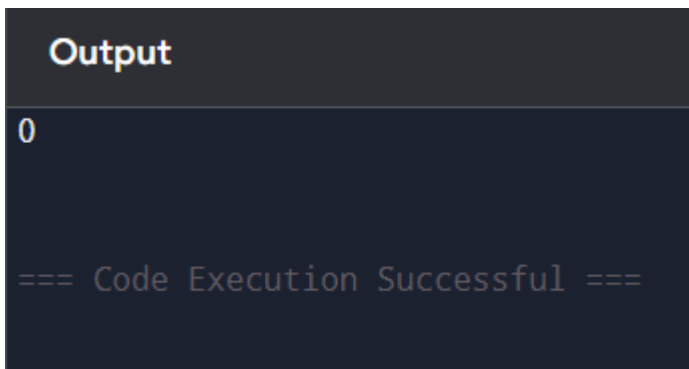
```
#include <iostream>
```

```

#include <queue>
using namespace std;
int countStudents(queue<int> students, queue<int> sandwiches) {
    int count = 0;
    while (!students.empty() && count < students.size()) {
        if (students.front() == sandwiches.front()) {
            students.pop();
            sandwiches.pop();
            count = 0;
        } else {
            students.push(students.front());
            students.pop();
            count++;
        }
    }
    return students.size();
}
int main() {
    queue<int> students, sandwiches;
    students.push(1); students.push(1); students.push(0); students.push(0);
    sandwiches.push(0); sandwiches.push(1); sandwiches.push(0); sandwiches.push(1);
    cout << countStudents(students, sandwiches) << endl; // Output: 0
    return 0;
}

```

Output:



```

Output
0

=== Code Execution Successful ===

```

Ques 3:

Given a circular integer array nums (i.e., the next element of nums[nums.length - 1] is nums[0]), return the next greater number for every element in nums.

The next greater number of a number x is the first greater number to its traversing-order next in the array, which means you could search circularly to find its next greater number. If it doesn't exist, return -1 for this number.

Code:

```
#include <iostream>
#include <stack>
using namespace std;
void nextGreater(int nums[], int n) {
    int res[n];
    stack<int> s;
    for (int i = 2 * n - 1; i >= 0; i--) {
        while (!s.empty() && s.top() <= nums[i % n]) {
            s.pop();}
        res[i % n] = s.empty() ? -1 : s.top();
        s.push(nums[i % n]);}
    for (int i = 0; i < n; i++) {
        cout << res[i] << " ";}}
int main() {
    int nums[] = {1, 2, 1};
    int n = 3;
    nextGreater(nums, n); // Output: 2 -1 2
    return 0;}
```

Output:

```
Output
2 -1 2

=== Code Execution Successful ===
```

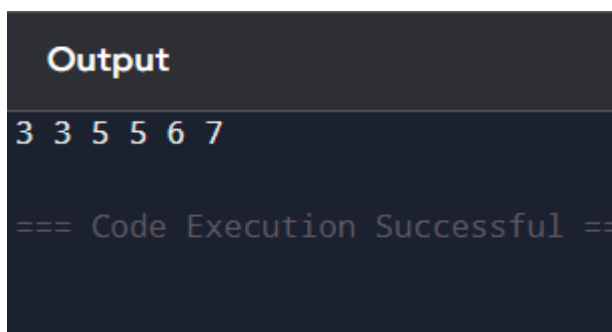
Ques 4:

You are given an array of integers `nums`, there is a sliding window of size `k` which is moving from the very left of the array to the very right. You can only see the `k` numbers in the window. Each time the sliding window moves right by one position.

Code:

```
#include <iostream>
#include <deque>
using namespace std;
void maxSlidingWindow(int nums[], int n, int k) {
    deque<int> dq;
    for (int i = 0; i < n; i++) {
        if (!dq.empty() && dq.front() == i - k) {
            dq.pop_front();
        }
        while (!dq.empty() && nums[dq.back()] < nums[i]) {
            dq.pop_back();
        }
        dq.push_back(i);
        if (i >= k - 1) {
            cout << nums[dq.front()] << " ";
        }
    }
}
int main() {
    int nums[] = {1, 3, -1, -3, 5, 3, 6, 7};
    int n = 8, k = 3;
    maxSlidingWindow(nums, n, k); // Output: 3 3 5 5 6 7
    return 0;}
```

Output:



The screenshot shows a dark-themed terminal window. At the top, the word "Output" is displayed in a light blue font. Below it, the numbers "3 3 5 5 6 7" are printed in a light blue font. At the bottom, the text "=== Code Execution Successful ===" is displayed in a light blue font.

Ques 5:

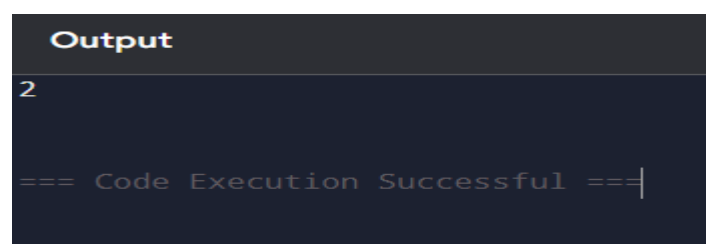
There are a number of plants in a garden. Each of the plants has been treated with some amount of pesticide. After each day, if any plant has more pesticide than the plant on its left, being weaker than the left one, it dies.

You are given the initial values of the pesticide in each of the plants. Determine the number of days after which no plant dies, i.e. the time after which there is no plant with more pesticide content than the plant to its left.

Code:

```
#include <iostream>
#include <stack>
using namespace std;
int poisonousPlants(int p[], int n) {
    stack<pair<int, int>> s;
    int days = 0;
    for (int i = 0; i < n; i++) {
        int maxDays = 0;
        while (!s.empty() && s.top().first >= p[i]) {
            maxDays = max(maxDays, s.top().second);
            s.pop();
        }
        maxDays = s.empty() ? 0 : maxDays + 1;
        days = max(days, maxDays);
        s.push({p[i], maxDays});
    }
    return days;
}
int main() {
    int p[] = {6, 5, 8, 4, 7, 10, 9};
    int n = 7;
    cout << poisonousPlants(p, n) << endl; // Output: 2
    return 0;
}
```

Output:



The screenshot shows a dark-themed window titled "Output". Inside, the number "2" is displayed on the first line. On the second line, the text "=== Code Execution Successful ===" is shown, followed by a vertical cursor bar.

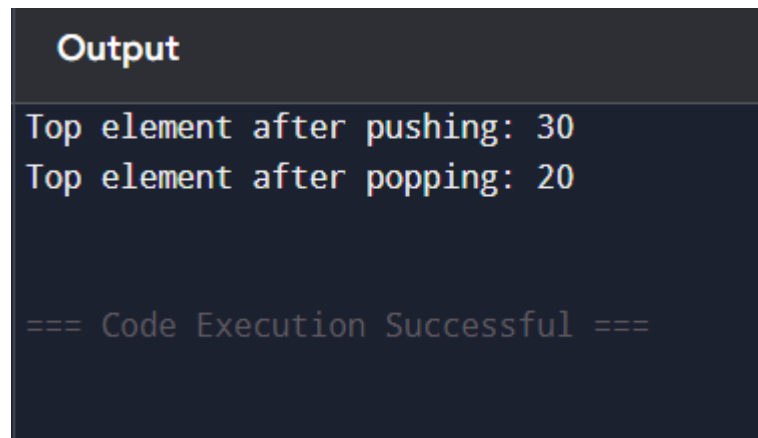
Ques 6:

To demonstrate basic stack operations in C++: pushing, accessing the top element, and popping elements.

Code:

```
#include <iostream>
#include <stack>
using namespace std;
int main() {
    stack<int> myStack;
    myStack.push(10);
    myStack.push(20);
    myStack.push(30);
    cout << "Top element after pushing: " << myStack.top() << endl;
    myStack.pop();
    cout << "Top element after popping: " << myStack.top() << endl;
    return 0;
}
```

Output:

A screenshot of a terminal window with a dark background. The title bar at the top says "Output" in white. The terminal shows the output of the C++ program: "Top element after pushing: 30" and "Top element after popping: 20" on separate lines. At the bottom, there is a green text message: "=== Code Execution Successful ===".

```
Output
Top element after pushing: 30
Top element after popping: 20

=== Code Execution Successful ===
```

Ques 7:

To find the first non-repeating character in a given string using a queue and frequency array in C++.

Code:

```
#include <iostream>
#include <string>
```

```

#include <queue>

using namespace std;

char findFirstNonRepeating(const string &str) {
    int freq[256] = {0};
    queue<char> q;
    for (char c : str) {
        freq[c]++;
        q.push(c);
        while (!q.empty() && freq[q.front()] > 1) {
            q.pop();
        }
    }
    return q.empty() ? -1 : q.front();
}

int main() {
    string input;
    cout << "Enter a string: ";
    cin >> input;
    char result = findFirstNonRepeating(input);
    if (result != -1) {
        cout << "First non-repeating character: " << result << endl;
    } else {
        cout << "-1" << endl;
    }
    return 0;
}

```

Output:

```

Output
Enter a string: abaa
First non-repeating character: b

=== Code Execution Successful ===

```


Ques 8:

Implement stack using array

Code:

```
#include <iostream>

using namespace std;

class Stack {

private:

    int arr[100]; // Array to hold stack elements

    int top;      // Index of the top element

public:

    Stack() { top = -1; } // Initialize the stack with an empty state

    bool isEmpty() {

        return top == -1; // Stack is empty if top is -1

    }

    bool isFull() {

        return top == 99; // Stack is full if top is 99 (size limit 100)}

    void push(int x) {

        if (isFull()) {

            cout << "Stack Overflow" << endl;

            return;}

        arr[++top] = x; // Increment top and insert the element

        cout << x << " pushed to stack" << endl;}

    void pop() {

        if (isEmpty()) {

            cout << "Stack Underflow" << endl;

            return;

        }

        cout << arr[top--] << " popped from stack" << endl; // Pop the top element

    }

}
```

```

int peek() {
    if (isEmpty()) {
        cout << "Stack is empty" << endl;
        return -1;
    }

    return arr[top]; // Return the top element without removing it
}

};

int main() {
    Stack stack;

    stack.push(10); // Push elements
    stack.push(20);
    stack.push(30);

    cout << "Top element is: " << stack.peek() << endl;

    stack.pop();    // Pop elements
    stack.pop();

    cout << "Top element is: " << stack.peek() << endl;

    return 0;
}

```

Output:

Output
10 pushed to stack
20 pushed to stack
30 pushed to stack
Top element is: 30
30 popped from stack
20 popped from stack
Top element is: 10
=== Code Execution Successful ===

Ques 9:

Implement Stack using Linked list .

Code:

```
#include <iostream>

using namespace std;

class Node {
public:
    int data;
    Node* next;
};

class Stack {
private:
    Node* top;
public:
    Stack() : top(nullptr) {}

    void push(int x) {
        Node* newNode = new Node();
        newNode->data = x;
        newNode->next = top;
        top = newNode;
    }

    void pop() {
        if (top != nullptr) {
            Node* temp = top;
            top = top->next;
            delete temp;
        }
    }
}
```

```

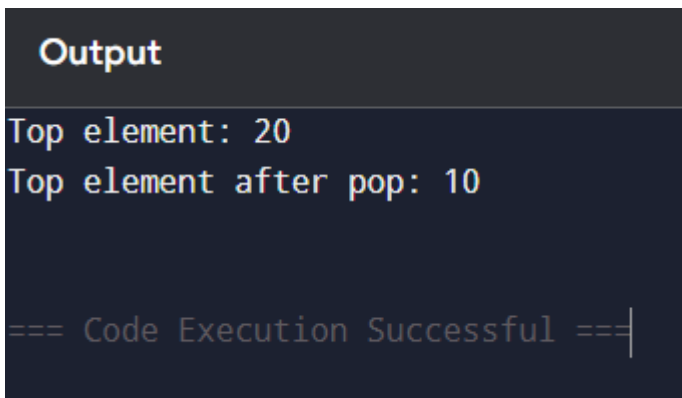
int peek() {
    return (top != nullptr) ? top->data : -1;
}

};

int main() {
    Stack stack;
    stack.push(10);
    stack.push(20);
    cout << "Top element: " << stack.peek() << endl; // Output: 20
    stack.pop();
    cout << "Top element after pop: " << stack.peek() << endl; // Output: 10
    return 0;
}

```

Output:



The screenshot shows a dark-themed window titled "Output". It contains the following text:

```

Top element: 20
Top element after pop: 10

=== Code Execution Successful ===

```

Ques 10:

Implement stack using 2 Queue.

Code:

```

#include <iostream>
#include <queue>
using namespace std;
class StackUsingQueues {
private:
    queue<int> q1, q2;

```

public:

```
void push(int x) {  
    // Push element into q1  
    q1.push(x);  
}  
void pop() {  
    if (q1.empty()) {  
        cout << "Stack is empty" << endl;  
        return;  
    }  
    while (q1.size() > 1) {  
        q2.push(q1.front());  
        q1.pop();  
    }  
    q1.pop();  
    swap(q1, q2);  
}  
int top() {  
    if (q1.empty()) {  
        cout << "Stack is empty" << endl;  
        return -1;  
    }  
    while (q1.size() > 1) {  
        q2.push(q1.front());  
        q1.pop();  
    }  
    int topElement = q1.front();  
    q2.push(topElement);  
    swap(q1, q2);  
    return topElement;  
}
```

```
bool isEmpty() {
    return q1.empty();
}

};

int main() {
    StackUsingQueues stack;
    stack.push(10);
    stack.push(20);
    stack.push(30);
    cout << "Top element: " << stack.top() << endl; // Output: 30
    stack.pop();
    cout << "Top element after pop: " << stack.top() << endl; // Output: 20
    stack.pop();
    stack.pop();
    cout << "Stack empty? " << (stack.isEmpty() ? "Yes" : "No") << endl; // Output: Yes

    return 0;
}
```

Output:

Output

```
Top element: 30
Top element after pop: 20
Stack empty? No
```