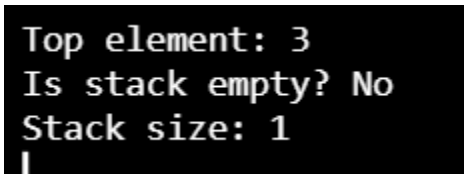
#1

```
#include <iostream>
#include <stack>
int main() {
    std::stack<int> S;

    S.push(5);
    S.push(3);

    std::cout << "Top element: " << S.top() << std::endl; // 3
    S.pop();
    std::cout << "Is stack empty? " << (S.empty() ? "Yes" : "No") << std::endl;
    std::cout << "Stack size: " << S.size() << std::endl;
}
```



#2

```
#include <iostream>
using namespace std;
class Stack {
    int topIndex;
    int arr[100]; // Fixed size for simplicity
    public:
        Stack() : topIndex(-1) {}
        // Push an element onto the stack
        void push(char x) {
            if (topIndex >= 99) {
            cout << "Stack overflow!" << endl;
            return;
            }
            arr[++topIndex] = x;
        }
        // Pop the top element from the stack
        void pop() {
            if (topIndex == -1) {
                cout << "Stack underflow!" << endl;
                return;
            }
```

```
        --topIndex;
        }
    // Return the top element of the stack
    char top() {
        if (topIndex == -1) {
            cout << "Stack is empty!" << endl;
            return -1;
        }
        return arr[topIndex];
    }
    // Return the size of the stack
    int size() {
        return topIndex + 1;
    }
    // Check if the stack is empty
    bool empty() {
        return topIndex == -1;
    }
};
int main() {
Stack S;
// Push elements onto the stack
S.push('N');
S.push('U');
// Print the current size
cout << "Stack size: " << S.size() << endl; // 2
// Push another element
S.push('R');
// Get the top element
cout << "Top element: " << S.top() << endl; // 'R'
// Pop the top element
S.pop();
cout << "Top element after pop: " << S.top() << endl; // 'U'
}
```

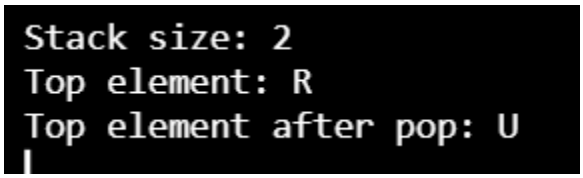
```
Stack size: 2
Top element: R
Top element after pop: U
```

#3

```
#include <iostream>
using namespace std;
class Node {
```

```
    public:
        char data;
        Node* next;
        Node(char d) : data(d), next(nullptr) {}
};
class Stack {
    Node* topNode;
    int stackSize; // To track the size of the stack
    public:
        Stack() : topNode(nullptr), stackSize(0) {}
        // Push an element onto the stack
        void push(char x) {
            Node* newNode = new Node(x);
            newNode->next = topNode;
            topNode = newNode;
            stackSize++;
        }
        // Pop the top element from the stack
        void pop() {
            if (!topNode) {
                cout << "Stack underflow!" << endl;
                return;
            }
            Node* temp = topNode;
            topNode = topNode->next;
            delete temp;
            stackSize--;
        }
        // Return the top element of the stack
        char top() {
            if (!topNode) {
                cout << "Stack is empty!" << endl;
                return -1;
            }
            return topNode->data;
        }
        // Return the size of the stack
        int size() {
            return stackSize;
        }
        // Check if the stack is empty
        bool empty() {
            return topNode == nullptr;
        }
```

```
};
int main() {
    Stack S;
    // Push elements onto the stack
    S.push('N');
    S.push('U');
    // Print the current size
    cout << "Stack size: " << S.size() << endl; // 2
    // Push another element
    S.push('R');
    // Get the top element
    cout << "Top element: " << S.top() << endl; // 'R'
    S.pop(); // Pop the top element
    cout << "Top element after pop: " << S.top() << endl; // 'U'
}
```



#4

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

template<typename T>

void reverseArray(T arr[], int n) {
    stack<T> stack;
    for (int i = 0; i < n; ++i)
        stack.push(arr[i]);
    for (int i = 0; i < n; ++i) {
        arr[i] = stack.top();
        stack.pop();
    }
}
int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int n = 5;
    reverseArray(arr, n);
    for (int i = 0; i < n; ++i)
```

```
    cout << arr[i] << " ";
}
```