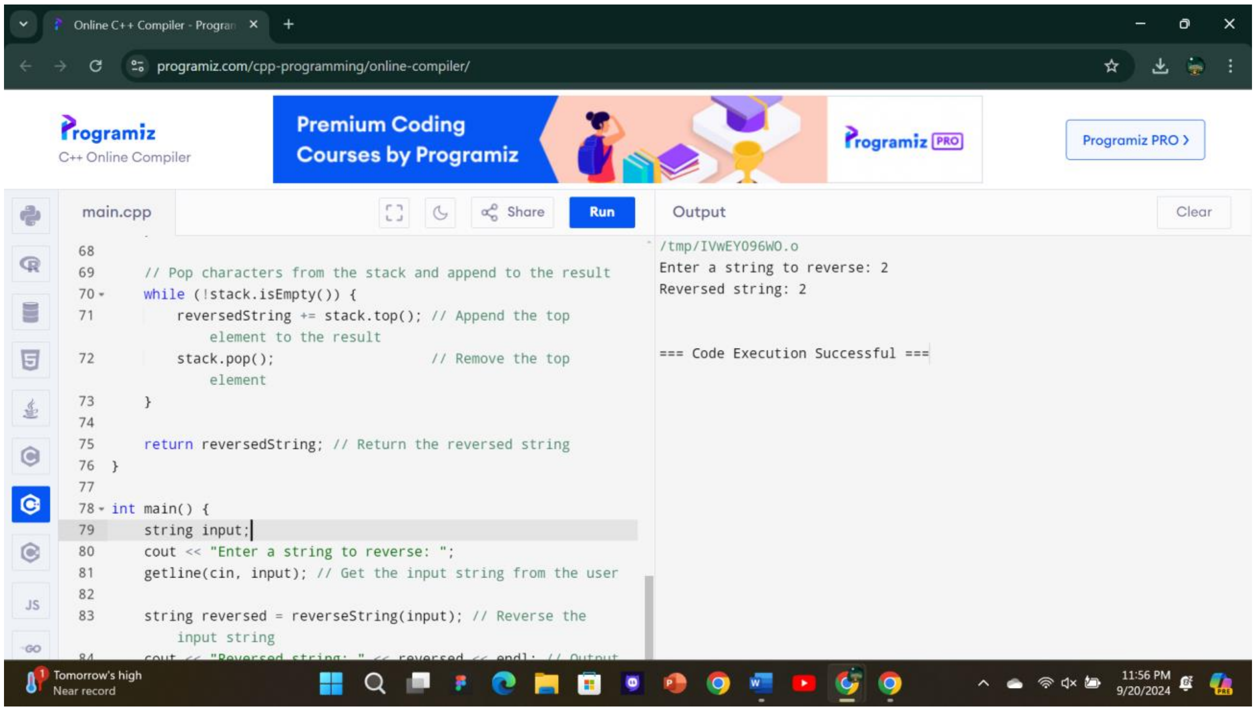
Name :- Haider Ali

SAP ID :- 53109

Programme :- BSCS 3-1

Q1

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

class BasicStack {
private:
    char* stackArray; // Poorly named variable for the stack array
    int topIndex;     // Tracks the top element's index
    int maxSize;      // Capacity of the stack

public:
    // Constructor with fixed capacity (inefficient)
    BasicStack(int size = 100) {
        stackArray = new char[size]; // No error handling for memory
allocation
        topIndex = -1;              // Stack is initially empty
        maxSize = size;              // Maximum size of the stack
    }

    // Destructor for freeing memory (basic and mandatory)
    ~BasicStack() {
        delete[] stackArray; // Cleans up the dynamic array
    }

    // Push elements onto the stack (no error handling for invalid inputs)
    void push(char element) {
        if (topIndex >= maxSize - 1) {
            cout << "Stack overflow!" << endl; // Simple overflow message
        } else {
            topIndex++;              // Increment top index
            stackArray[topIndex] = element; // Add element to stack
        }
    }

    // Pop the top element (doesn't return the popped element)
    void pop() {
        if (isEmpty()) {
            cout << "Stack underflow!" << endl; // Simple underflow
message
        } else {
            topIndex--; // Just decrements the index
        }
    }

    // Peek at the top element (no bounds checking)
    char top() {
        if (!isEmpty()) {
            return stackArray[topIndex]; // Return the top element
        } else {
            cout << "Stack is empty!" << endl;
            return '\0'; y
        }
    }
    bool isEmpty() {
        return topIndex == -1; // Stack is empty if topIndex is -1
    }
};

string reverseString(const string& input) {
    BasicStack stack(input.length()); // Create a stack based on the
string length
    string reversedString = "";       // String to hold the reversed result

    // Push all characters of the string onto the stack
    for (int i = 0; i < input.length(); i++) {
        stack.push(input[i]);
    }

    // Pop characters from the stack and append to the result
    while (!stack.isEmpty()) {
        reversedString += stack.top(); // Append the top element to the
result
        stack.pop();                   // Remove the top element
    }

    return reversedString; // Return the reversed string
}

int main() {
    string input;
    cout << "Enter a string to reverse: ";
    getline(cin, input); // Get the input string from the user

    string reversed = reverseString(input); // Reverse the input string
    cout << "Reversed string: " << reversed << endl; // Output the
reversed string

    return 0;
}
```



```cpp
    // Pop characters from the stack and append to the result
    while (!stack.isEmpty()) {
        reversedString += stack.top(); // Append the top element to the result
        stack.pop();                   // Remove the top element
    }

    return reversedString; // Return the reversed string
}

int main() {
    string input;
    cout << "Enter a string to reverse: ";
    getline(cin, input); // Get the input string from the user

    string reversed = reverseString(input); // Reverse the input string
    cout << "Reversed string: " << reversed << endl; // Output
```

```
/tmp/IVwEY096WO.o
Enter a string to reverse: 2
Reversed string: 2


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