Name :- Ahmed Ali Asif

SAP ID :- 55346

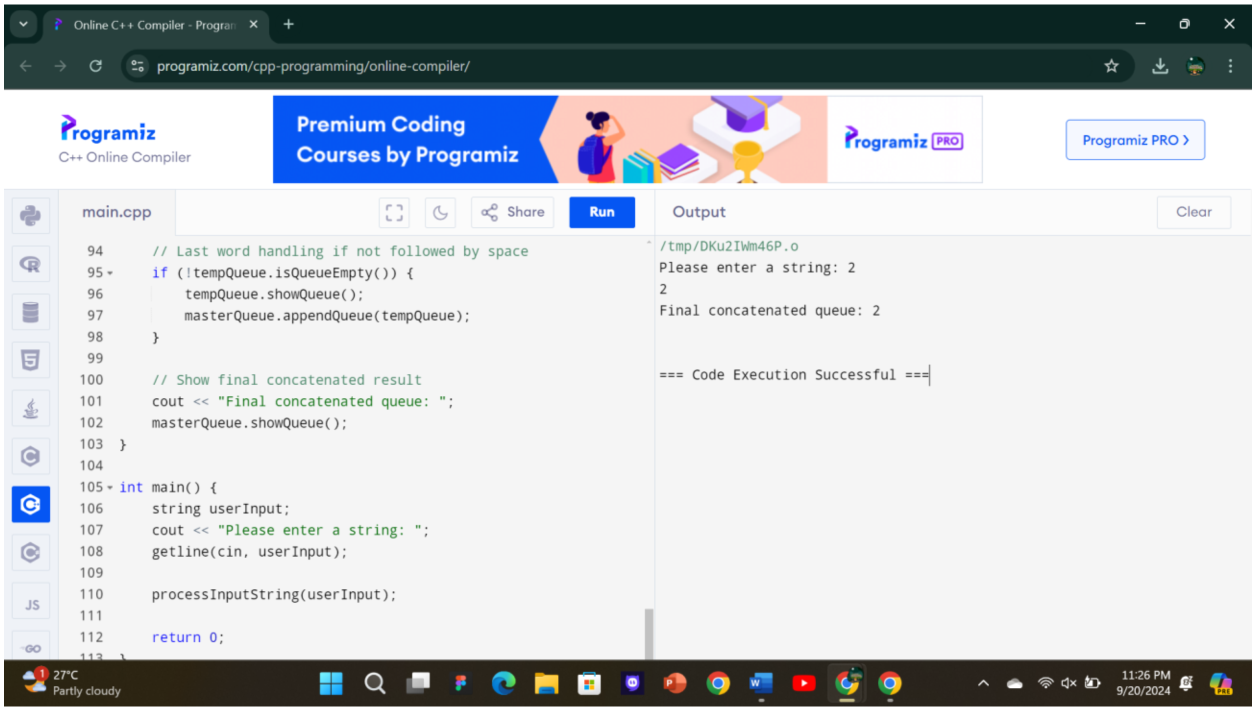
Programme :- BSCS 3-1

Assignment #4

Q1

```
#include <iostream>

#include <string>

using namespace std;


class SimpleQueue {

private:

    char* dataArray;

    int frontIndex;

    int rearIndex;

    int maxSize;

    int itemCount;


public:

    SimpleQueue(int size = 100) {

        dataArray = new char[size];

        maxSize = size;

        frontIndex = 0;

        rearIndex = -1;

        itemCount = 0;
    }
    ~SimpleQueue() {
        delete[] dataArray;
    }

    // Enqueue method with verbose conditions
    void insert(char element) {
        if (itemCount == maxSize) {
            cout << "Error: Queue is full!" << endl;
        } else {
            rearIndex = (rearIndex + 1) % maxSize; // Modulo logic without
explanation
            dataArray[rearIndex] = element;
            itemCount++;
        }
    }

    // Dequeue method that handles underflow poorly
    char remove() {
        if (isQueueEmpty()) {
            cout << "Queue is empty, returning null!" << endl;
            return '\0'; // Returning null char instead of handling it properly
        } else {
            char value = dataArray[frontIndex];
            frontIndex = (frontIndex + 1) % maxSize;
            itemCount--;
            return value;
        }
    }

    // Method to check if the queue is empty
    bool isQueueEmpty() {
        return itemCount == 0;
    }

    // Display function with a verbose loop and logic
    void showQueue() {
        if (isQueueEmpty()) {
            cout << "Queue is currently empty." << endl;
        } else {
            int i = frontIndex;
            for (int count = 0; count < itemCount; count++) {
                cout << dataArray[i] << " ";
                i = (i + 1) % maxSize;
            }
            cout << endl;
        }
    }

    // Concatenation method using a loop (inefficient and verbose)
```

```
    void appendQueue(SimpleQueue& q) {
        while (!q.isQueueEmpty()) {
            insert(q.remove()); // Inserting one by one from another queue
        }
    }
};

void processInputString(string input) {
    SimpleQueue masterQueue(500); // Large default size without
justification
    SimpleQueue tempQueue;
    string tempWord = ""; // Unnecessary string variable

    for (char ch : input) {
        if (ch != ' ') {
            tempQueue.insert(ch); // Insert characters until space
        } else {
            tempQueue.showQueue(); // Show each queue before
concatenation
            masterQueue.appendQueue(tempQueue);
            tempQueue = SimpleQueue(); // Reinitialize the queue
(amateurish)
        }
    }
    // Last word handling if not followed by space
    if (!tempQueue.isQueueEmpty()) {
        tempQueue.showQueue();
        masterQueue.appendQueue(tempQueue);
    }

    // Show final concatenated result
    cout << "Final concatenated queue: ";
    masterQueue.showQueue();
}

int main() {
    string userInput;
    cout << "Please enter a string: ";
    getline(cin, userInput);

    processInputString(userInput);

    return 0;
}
```



```cpp
    // Last word handling if not followed by space
    if (!tempQueue.isQueueEmpty()) {
        tempQueue.showQueue();
        masterQueue.appendQueue(tempQueue);
    }

    // Show final concatenated result
    cout << "Final concatenated queue: ";
    masterQueue.showQueue();
}

int main() {
    string userInput;
    cout << "Please enter a string: ";
    getline(cin, userInput);

    processInputString(userInput);

    return 0;
```

```
/tmp/DKu2IWm46P.o
Please enter a string: 2
2
Final concatenated queue: 2


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