LAB#7

F24-0767

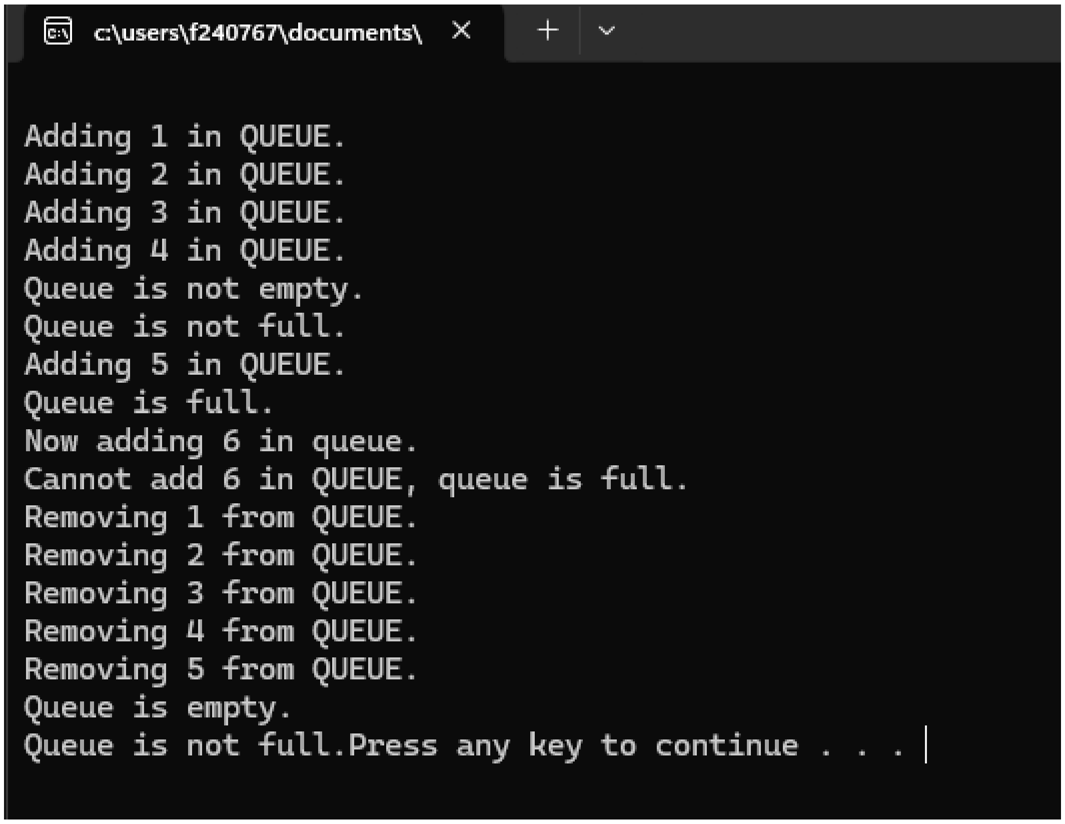
Ayaan Amer

TASK #1:

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

class Queue {
private:
	int* arr;
	int size;
	int front;
	int rear;
	int element;

public:
	Queue(int size) {
		this->size = size;
		arr = new int[size];
		front = rear = -1;
		element = 0;
		for (int i = 0; i < size; i++) {
			arr[i] = 0;
		}
	}

	bool isEmpty() {

		return (front == -1 && rear == -1);
	}

	bool isFull() {

		return (element == size);
	}

	void enQue(int value) {
		if (isFull()) {
			cout << "\nCannot add " << value << " in QUEUE, queue is full.";
			return;
		}
		if (isEmpty()) {
			front = rear = 0;
		}
		else {
			rear = (rear + 1) % size;
		}
		arr[rear] = value;
```

```
			element++;
			cout << "\nAdding " << value << " in QUEUE.";
		}

		int deQue() {
			if (isEmpty()) {
				cout << "\nQueue is empty, cannot dequeue (underflow).";
				return -1;
			}
			int temp = arr[front];
			cout << "\nRemoving " << temp << " from QUEUE.";
			arr[front] = 0;
			if (front == rear) {

				front = rear = -1;
			}
			else {
				front = (front + 1) % size;
			}
			element--;
			return temp;
		}

		~Queue() {
			delete[] arr;
		}
	};

	int main() {
		Queue obj(5);

		obj.enQue(1);
		obj.enQue(2);
		obj.enQue(3);
		obj.enQue(4);

		if (!obj.isEmpty()) {
			cout << "\nQueue is not empty.";
		}
		if (!obj.isFull()) {
			cout << "\nQueue is not full.";
		}

		obj.enQue(5);

		if (obj.isFull()) {
			cout << "\nQueue is full.";
		}

		cout << "\nNow adding 6 in queue.";
		obj.enQue(6);

		obj.deQue();
		obj.deQue();
		obj.deQue();
		obj.deQue();
		obj.deQue();
```

```
        if (obj.isEmpty()) {
               cout << "\nQueue is empty.";
        }
        if (!obj.isFull()) {
               cout << "\nQueue is not full.";
        }

         system("pause");
        return 0;
}
```



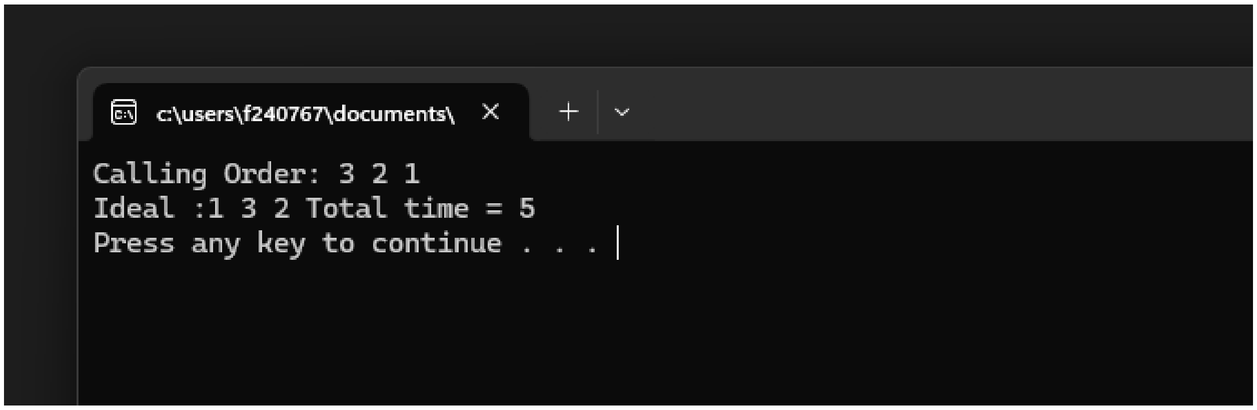
```
c:\users\f240767\documents\

Adding 1 in QUEUE.
Adding 2 in QUEUE.
Adding 3 in QUEUE.
Adding 4 in QUEUE.
Queue is not empty.
Queue is not full.
Adding 5 in QUEUE.
Queue is full.
Now adding 6 in queue.
Cannot add 6 in QUEUE, queue is full.
Removing 1 from QUEUE.
Removing 2 from QUEUE.
Removing 3 from QUEUE.
Removing 4 from QUEUE.
Removing 5 from QUEUE.
Queue is empty.
Queue is not full.Press any key to continue . . .
```

TASK#2:

```
#include<iostream>

using namespace std;

int output;


class Queue {

public:

        int* arr;
```

```
int size;

int count;

int front;

int rear;

Queue() {

        size = 0;

        arr = nullptr;

        front = 0;

        rear = -1;

        count = 0;

}

void sizze(int s) {

        size = s;

        arr = new int[s];

        for (int i = 0; i < s; i++) arr[i] = -1;

        front = 0;

        rear = -1;

        count = 0;

}

bool isempty() {

        return count == 0;

}

void Enqueue(int v) {

        rear = (rear + 1) % size;

        arr[rear] = v;

        count++;

}
```

```
        int Dequeue() {

                output = arr[front];

                front = (front + 1) % size;

                count--;

                return output;

        }

        void display() {

                for (int i = 0; i < count; i++) {

                        int idx = (front + i) % size;

                        cout << arr[idx] << " ";

                }

                cout << endl;

        }

};

int main() {

        int n = 3;

        Queue Q;

        Q.sizze(n);

        Q.Enqueue(3);

        Q.Enqueue(2);

        Q.Enqueue(1);

        cout << "Calling Order: ";

        Q.display();

        int ideal[3] = { 1, 3, 2 };

        cout << "Ideal :";

        for (int i = 0; i < n; i++) {
                cout << ideal[i] << " ";

        }

        int idx = 0;

        int time = 0;


        while (!Q.isempty()) {

                if (Q.arr[Q.front] == ideal[idx]) {


                        Q.Dequeue();

                        idx++;

                        time++;

                }

                else {


                        Q.Dequeue();

                        Q.Enqueue(output);

                        time++;

                }

        }


        cout << "Total time = " << time << endl;

        system("pause");

        return 0;

}
```

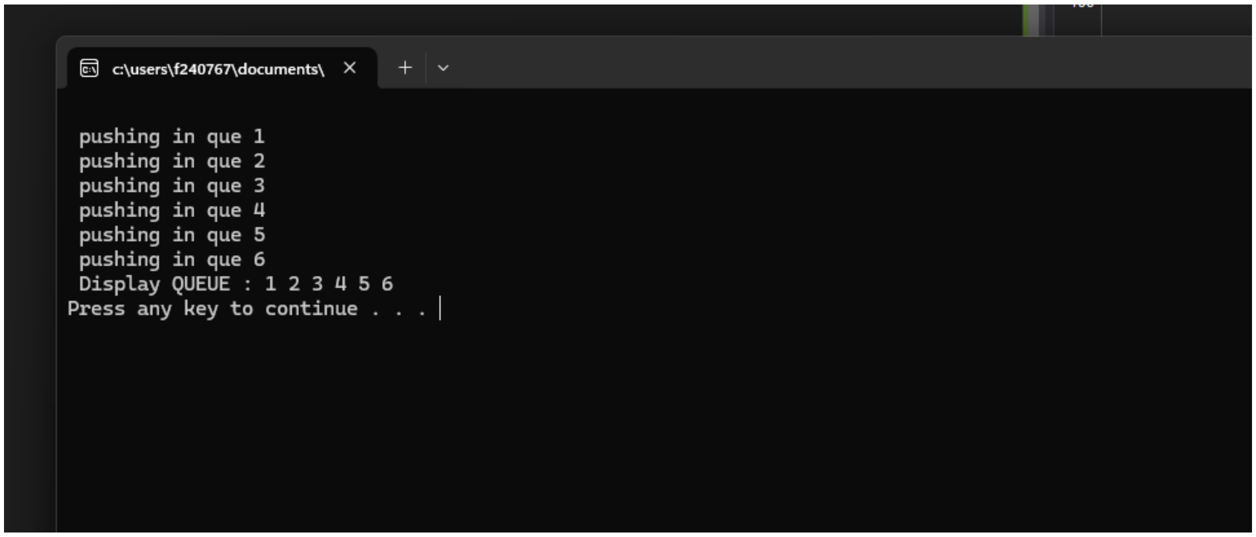


```
Calling Order: 3 2 1
Ideal :1 3 2 Total time = 5
Press any key to continue . . .
```

Task#3:

```
#include<stack>
#include<iostream>
using namespace std;
class queue {
private:
	stack<int> s1;
	stack<int> s2;
public:
	void enque(int value) {
		cout << "\n pushing in que " << value;
		s1.push(value);
	}
	bool isEmpty() {
		if (s1.empty())
			return true;
		return false;
	}
	int deque() {
		while (!s1.empty())
		{
			s2.push(s1.top());
			s1.pop();

		}
		int temp = s2.top();
		s2.pop();
		while (!s2.empty()) {
			s1.push(s2.top());
			s2.pop();
		}
		return temp;
	}
	void display() {
		cout << "\n Display QUEUE : ";
		while (!s1.empty())
		{
			s2.push(s1.top());
			s1.pop();

		}
```

```cpp
                while (!s2.empty()) {
                        cout << s2.top() << " ";
                        s1.push(s2.top());
                        s2.pop();
                }
                cout << "\n";
        }

};
int main() {
        queue obj;
        obj.enque(1);
        obj.enque(2);
        obj.enque(3);
        obj.enque(4);
        obj.enque(5);
        obj.enque(6);
        obj.display();
        system("pause");
        return 0;
}
```



```
 pushing in que 1
 pushing in que 2
 pushing in que 3
 pushing in que 4
 pushing in que 5
 pushing in que 6
 Display QUEUE : 1 2 3 4 5 6
Press any key to continue . . .
```