Table of Contents

Table of figures II

The C++ code 1

Pseudo code 6

Flow chart 7

GUI 8

Table of figures

Figure 1 : Flow Chart.....7

Figure 2 : GUI.....8

The C++ code :

```
#include <iostream>
#include <string>
#include <iomanip>
#include <cstdlib>
#include <ctime>
#include <vector>
using namespace std;
const int MAX_SIZE = 4;

struct Student
{
    string name;
    int id;
};

class CircularQueue
{
private:
    Student items[MAX_SIZE];
    int front, rear;
    static int nextID;

public:
    CircularQueue()
    {
        front = rear = -1;
    }

    bool isEmpty()
    {
        return front == -1;
    }

    bool isFull()
    {
        return (front == 0 && rear == MAX_SIZE - 1) || (rear == front - 1) || (front
== rear + 1);
    }
    void enqueue(const Student& newStudent)
    {
        if (isFull())
        {
            cout << "Queue Overflow" << endl;
        }
        else
        {
            if (front == -1)
```

```
            {
                front = rear = 0;
            }
            else if (rear == MAX_SIZE - 1 && front != 0)
            {
                rear = 0;
            }
            else
            {
                rear++;
            }
            items[rear] = newStudent;
            items[rear].id = nextID;
            nextID = (nextID + 1) % 10000;
        }
    }



    void dequeue()
    {
        if (isEmpty())
        {
            cout << "Queue Underflow" << endl;
        }
        else
        {
            if (front == rear)
            {
                front = rear = -1;
            }
            else if (front == MAX_SIZE - 1)
            {
                front = 0;
            }
            else
            {
                front++;
            }
        }
    }

    Student frontElement()
    {
        if (isEmpty())
        {
            cout << "Queue is empty" << endl;
            return { "", -1 };
        }
        else
        {
            return items[front];
```

```
        }
    }

    void display()
    {
        if (isEmpty())
        {
            cout << "Queue is empty" << endl;
        }
        else
        {
            cout << "+-------------------------------------------------------------+"
<< endl;
            cout << "|Name                                                  |ID     |"
<< endl;
            cout << "+-------------------------------------------------------------+"
<< endl;
            if (front <= rear)
            {
                for (int i = front; i <= rear; i++)
                {
                    cout << "| " << setw(49) << left << items[i].name << " | " <<
setw(5) << right << items[i].id << " |" << endl;
                }
            }
            else
            {
                for (int i = front; i < MAX_SIZE; i++)
                {
                    cout << "| " << setw(49) << left << items[i].name << " | " <<
setw(5) << right << items[i].id << " |" << endl;
                }
                for (int i = 0; i <= rear; i++)
                {
                    cout << "| " << setw(49) << left << items[i].name << " | " <<
setw(5) << right << items[i].id << " |" << endl;
                }
            }
            cout << "+-------------------------------------------------------------+"
<< endl;
        }
    }
};

int CircularQueue::nextID = 2210;

void Ch()
{
    vector<Student> registeredStudents;
    CircularQueue waitingList;
    int choice;
```

```
    do
    {
        cout << "Choose an option:" << endl;
        cout << "1. Add Student to Waiting List" << endl;
        cout << "2. Process Registration" << endl;
        cout << "3. Display Registered Students" << endl;
        cout << "4. Display Students in Waiting List" << endl;
        cout << "5. Exit" << endl;
        cout << "Enter choice: ";
        cin >> choice;
        system("cls");

        switch (choice)
        {
        case 1:
        {
            Student newStudent;
            if (waitingList.isFull())
            {
                cout << "Waiting List is Full"<<endl;
                break;
            }
            else
            {
                cout << "Enter student name: ";
                cin.ignore();
                getline(cin, newStudent.name);
                waitingList.enqueue(newStudent);
                cout << "Student " << newStudent.name <<" registered successfully."
<< endl;
                break;
            }
        }
        case 2:
        {
            if (!waitingList.isEmpty())
            {
                Student newStudent = waitingList.frontElement();
                waitingList.dequeue();
                registeredStudents.push_back(newStudent);
                cout << "Student " << newStudent.name << " with ID " <<
newStudent.id << " registered successfully." << endl;
            }
            else
            {
                cout << "No students in the waiting list." << endl;
            }
            break;
        }
        case 3:
        {
```

```
            cout << "+-----------------------------------------------------------+"
<< endl;
            cout << "|Name                                                 |ID     |"
<< endl;
            cout << "+-----------------------------------------------------------+"
<< endl;
            for (const auto& student : registeredStudents)
            {
                cout << "| " << setw(49) << left << student.name << " | " << setw(5)
<< right << student.id << " |" << endl;
            }
          cout << "+-----------------------------------------------------------+" <<
endl;
            break;
        }
        case 4:
            waitingList.display();
            break;
        case 5:
            cout << "Exiting program..." << endl;
            break;
        default:
            cout << "Invalid choice. Please try again." << endl;
        }
    }
    while (choice != 5);
}

int main()
{
    Ch();
    return 0;
}
```

Pseudo code :

We have a MAX_SIZE queue size of 4
Each student enter your name
We have a circular queue of students
    Check if it's empty
    Check if it's full
    Add a new student to the end of the queue
    Remove a student from the front of the queue
    Get the first student in the queue
    Show all students in the queue
Each student added will have a unique ID starting from 2210
We have a function called Ch that does the following
      Keeps track of registered students in a list
      Manages a waiting list of students
      Repeats until the user decides to exit:
      Provides options for the user to choose from
      Reads the user's choice
      Depending on the choice
          Adds a student to the waiting list
          Registers a student if there are any in the waiting list
          Shows all registered students
          Shows all students in the waiting list
          Exits the program
  we have a main function that starts the program by calling Ch

Flow chart :

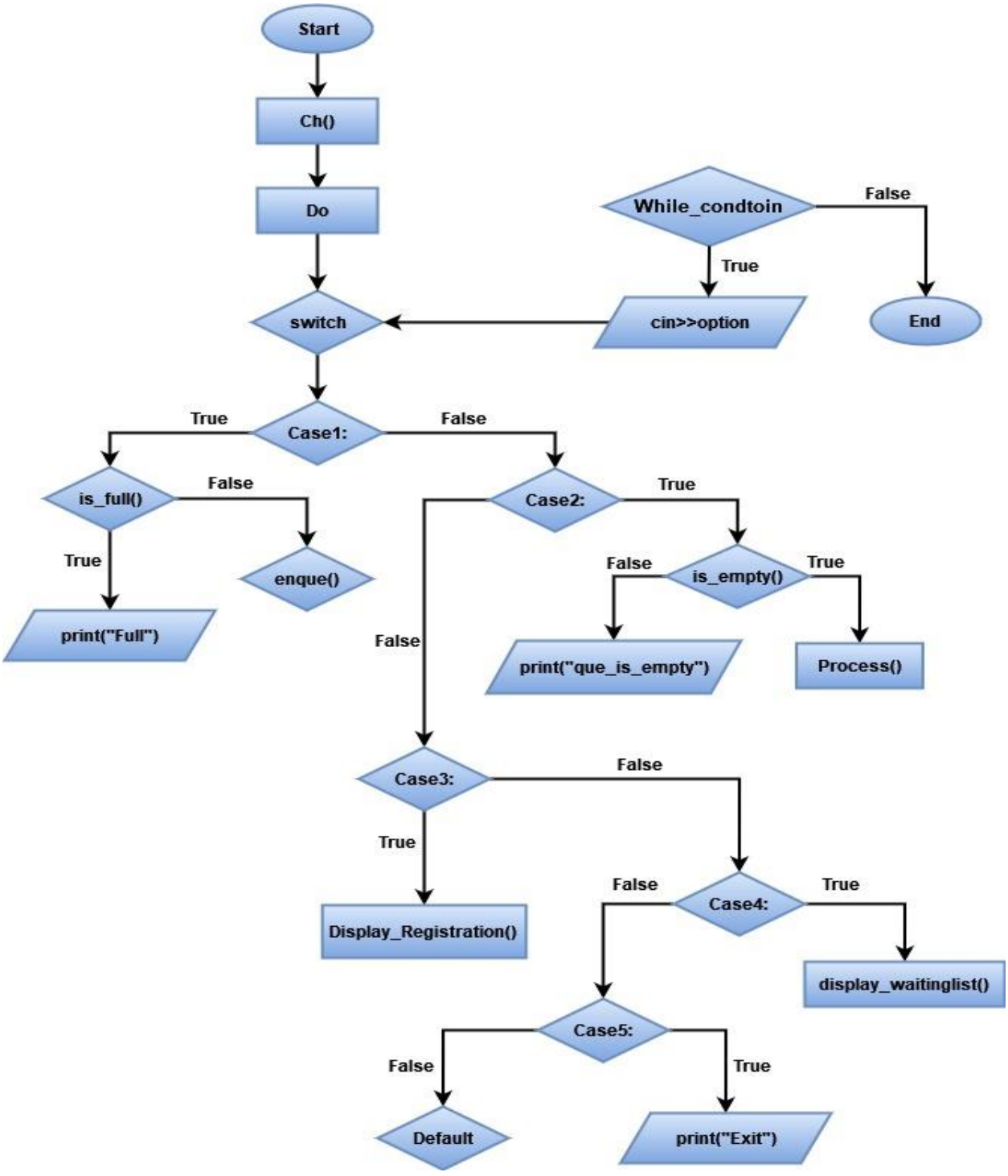

Figure1:Flow Chart

GUI :

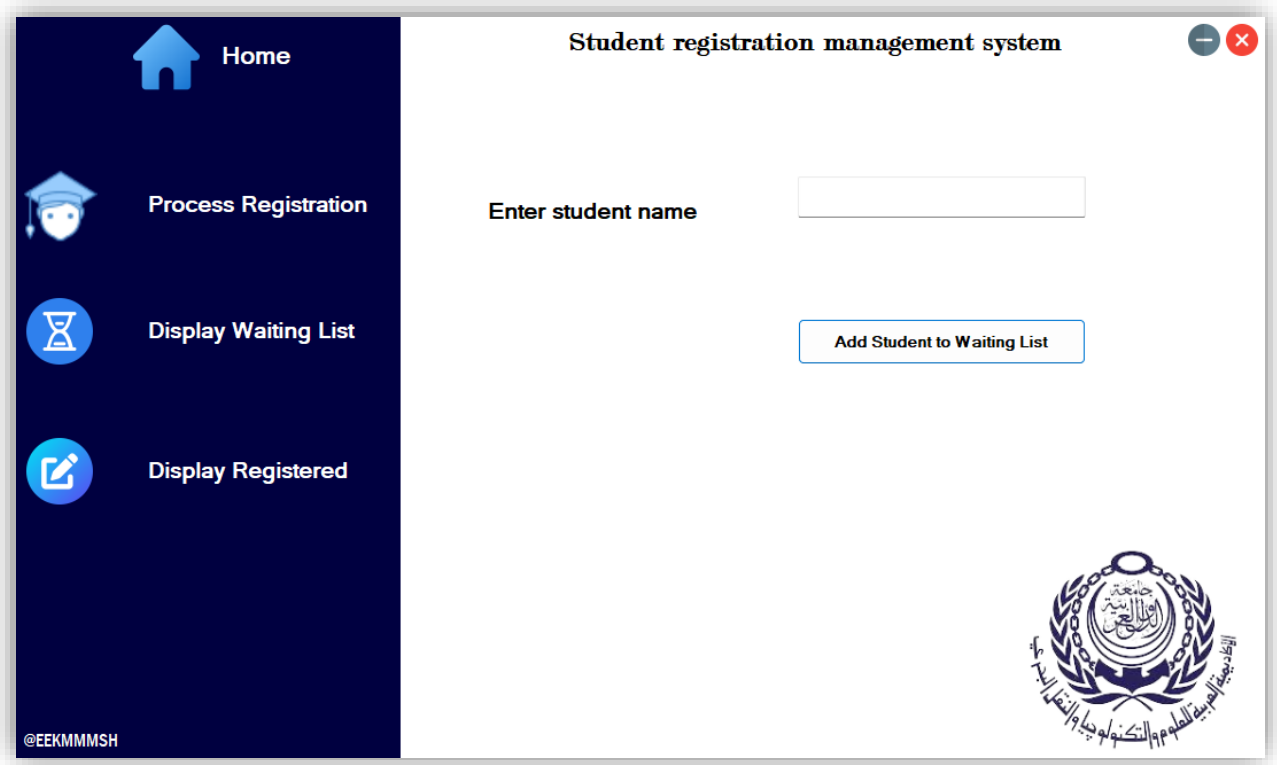

Figure2:GUI