# Department of Computing

# School of Electrical Engineering and Computer Science

**CS-250: Data Structure and Algorithms**

**Class: BESE 13AB**

# 

# Lab 5: Implmentation of Queues in different problems

**Date: 20th October, 2023**

**Time: 10 am - 1 pm & 2 pm - 5 pm**

# Lab Instructor: Anum Asif

# 

# Lab 5: Implmentation of Queues in different problems

**Lab Tasks**

You have to implement a waiting room management system in an emergency ward of a hospital. Your program will assign an Id number to a patient in a first come first serve basis. The lower the id, the sooner the service will be provided to the patient.

Your program will contain the following methods:

**RegisterPatient():** This method assigns an Id (which is auto-generated) to a patient and register him/her to the system.

**ServePatient():** This method calls a patient to provide hospital service to him/her.

**CancelAll():** This method cancels all appointments of the patients so that the doctor can go to lunch.

**CanDoctorGoHome():** This method returns true if no one is waiting, otherwise, returns false.

**ShowAllPatient():** This method shows all ids of the waiting patients in SORTED order. (Hint: use the sorting methods learnt in class using the appropriate data-structure for each task) [Sorted according to their names]

**Solution:**

|  |
| --- |
| Solution |
| #include <iostream>  #include <string>  #include <vector>  #include <algorithm>  using namespace std;  class hospital\_management\_system  {  private:      struct Patients      {          string name;          int id;          bool operator<(const Patients &other) const          {              return name < other.name;          }      };      Patients \*queue\_patients; // Dynamic array to store patients      int size;               // Current number of patients in the queue      int capacity;           // Maximum capacity of the queue      int front;              // Index of the front (head) of the queue      int rear;               // Index of the rear (end) of the queue      int current\_id;         // Counter for generating patient IDs  public:      hospital\_management\_system(int a) : capacity(a), size(0), rear(-1), front(0), current\_id(1)      {          queue\_patients = new Patients[capacity]; // Initialize the patient queue      };      void register\_patient(string const &name)      {          if (size < capacity)          {              Patients new\_patient = {name, current\_id}; // Create a new patient              rear = (rear + 1) % capacity; // Update the rear index              queue\_patients[rear] = new\_patient; // Add the new patient to the queue              current\_id++; // Increment the ID counter              size++;      // Increment the queue size              cout << "\nNew Patient with name: " << new\_patient.name << " has been registered with id: " << new\_patient.id;          }          else          {              cout << "\n The queue is already full, please wait !";          }      }      void serve\_patient()      {          if (size > 0)          {              Patients next\_patient = queue\_patients[front]; // Get the next patient to serve              cout << "Patient number: " << next\_patient.id << " please come forward for your turn.";              front = (front + 1) % capacity; // Update the front index              size--;                        // Decrement the queue size          }          else          {              cout << "\n No patient to be served at the moment!";          }      }      void cancel\_all()      {          front = -1;   // Reset the front index          rear = -1;    // Reset the rear index          size = 0;     // Reset the queue size          cout << "\n The doctor has gone for lunch, please wait till the doctor returns.";      }      bool can\_doctor\_go\_home()      {          return size == 0; // Check if there are no patients in the queue      }      void show\_all\_patient()      {          if (size == 0)          {              cout << "\nNo Patients Waiting.";              return;          }          else          {              Patients \*sorted\_patients = new Patients[size]; // Create an array to store sorted patients              int sorted\_index = 0;              for (int i = front; i <= rear; i++)              {                  sorted\_patients[sorted\_index++] = queue\_patients[i]; // Copy patients into the sorted array              }              std::sort(sorted\_patients, sorted\_patients + size, [](const Patients &a, const Patients &b)                       { return a.name < b.name; }); // Sort patients by name              cout << "Waiting patients in sorted order:" << std::endl;              for (int i = 0; i < size; i++)              {                  cout << "ID: " << sorted\_patients[i].id << ", Name: " << sorted\_patients[i].name << endl; // Display sorted patients              }              delete[] sorted\_patients; // Clean up memory          }      }  };  int main()  {      hospital\_management\_system hospital(5); // Initialize the system with a capacity of 5      hospital.register\_patient("Kashif");      hospital.register\_patient("Bob");      hospital.register\_patient("Charlie");      hospital.register\_patient("Faizan");      hospital.register\_patient("Ryan");      cout << "\nInitial Queue:" << endl;      hospital.show\_all\_patient();      cout << endl;      hospital.serve\_patient();      hospital.serve\_patient();      cout << "\nQueue After Serving Two Patients:" << endl;      hospital.show\_all\_patient();      cout << "\nDoctor can go home? ";      if (hospital.can\_doctor\_go\_home())      {          cout << "Yes" << endl;      }      else      {          cout << "No" << endl;      }      cout << endl;      hospital.cancel\_all();      cout << "\nQueue After Canceling All Appointments:" << endl;      hospital.show\_all\_patient();      return 0;  } |