**Hospital Management System**

**By**

**Saifullah Anik  
ID: 221-15-4922**

**Radoanul Arifen  
ID: 221-15-5284**

**Md. Shakib Ahammed  
ID: 221-15-5431**

Supervised By

**Ms. Amatul Bushra Akhi**

Assistant Professor

Department of CSE

Daffodil International University



**DAFFODIL INTERNATIONAL UNIVERSITY**

**DHAKA, BANGLADESH**

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
|  |  |
|  |  |
| 1. Introduction | 3 |
| 2. Problem Statement | 3 |
| 3. Objectives | 3 |
| 4. System Architecture & Data Structures Used | 3 |
| 5. Implementation Details | 4 |
| 6. Code Implementation | 5 |
| 7. Conclusion | 14 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## 1. Introduction

## The Hospital Management System is designed for improving a hospital's daily operations. The purpose of the report is to present an overview of the project, which focuses on patient registration, serial management, and medicine store operations. All of these are implemented using C programming and data structures.

## In this report, we tried to summarize the work we did in our project using C programming and some basic Data Structures.

## 2. Problem Statement

The system's primary focus is on the effective and coordinated management of hospital tasks such as patient registration, serialization, and medicine distribution. The system ensures that patients are treated in an organized way and that their medical records are kept in order.

## 3. Objectives

The main objectives of this project are as follows:

* To maintain the record of patient information.
* To manage the patient queue efficiently using queue data structure.
* To handle pharmacy operations through a stack data structure.

**4. System Architecture & Data Structures Used**

The System contains three parts with three different data structures. They are:

**Patient Registration (Linked List):** We used linked list data structure so that the patient registration can be handled easily and smoothly. Linked list allows to add or delete any data from any position whereas we cannot do that in queue and stack. So, we used it for patients’ flexibility.

**Serial Management (Queue):** Now we used queue data structure here to ensure the first in, first out (FIFO) policy here. Because it will ensure the queue is maintained properly and first comer will get first service.

**Pharmacy Section (Stack):** We used stack operation here to maintain the latest medicine on top. Because in stack whatever we enter at last we get it on the top.

**5. Implementation Details**

**1. Patient Registration**

The system takes the following information from patients to register the person in the database:

* Unique ID
* Name
* Age
* Blood group
* Mobile number
* Patients are registered with unique IDs to ensure data integrity and easy access.
* Patients can be inserted and deleted from the linked list as needed.
* The system provides the ability to search for a specific patient using their unique ID, ensuring quick access to patient information.
* The system also allows to get the total number of patients registered in the whole system.

**2. Serial Management**

* Serialization of patients is managed using a queue data structure.
* Patients who need to be assigned a serial number must be registered first. This ensures that no patient is assigned a serial without prior registration.
* Patients have to input their unique ID to get the serial. The system follows the "first-come-first-served" principle, maintaining a fair and orderly queue.
* This functionality helps in maintaining a systematic order for patient treatment.

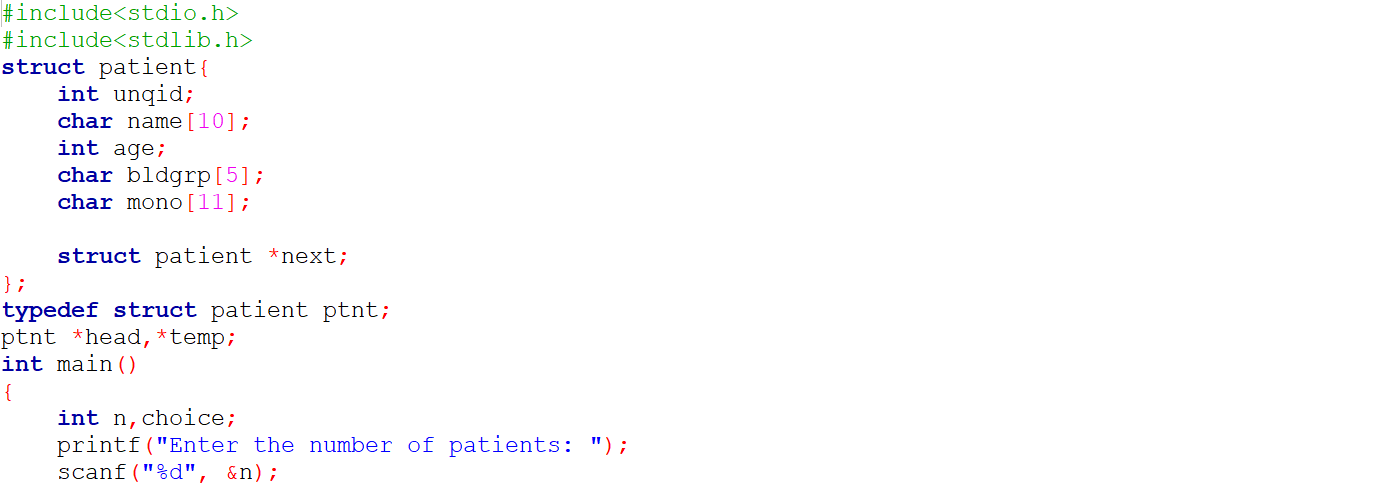
**3. Pharmacy Section**

* The medicine section is implemented using a stack data structure.
* Medications are added and removed from the top of the stack. New medications are placed on top of the existing ones.
* This method ensures that the most recent medication is quickly available for dispensing.
* The stack-based system provides an efficient way to manage medicine stock and ensures that patients receive the most recently prescribed medicines.

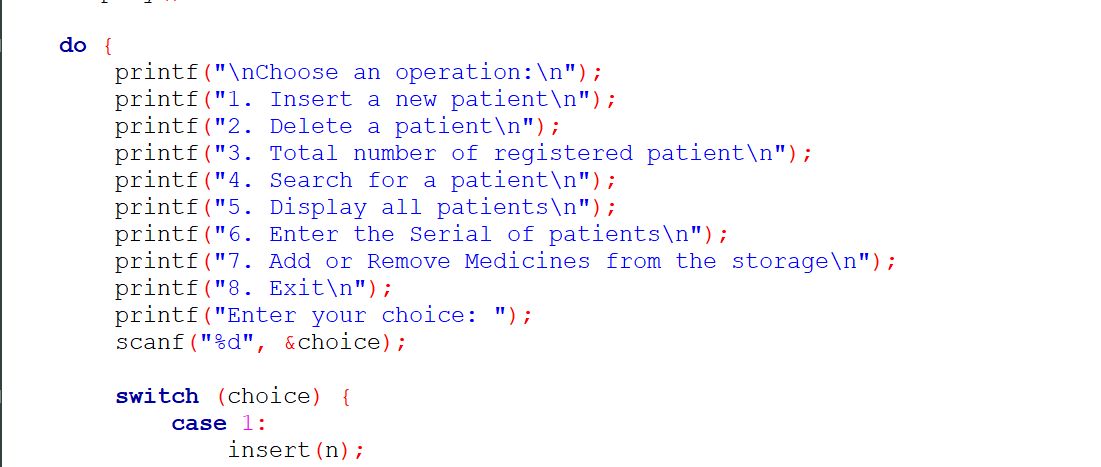
**6. Code Implementations**

**Source Code**

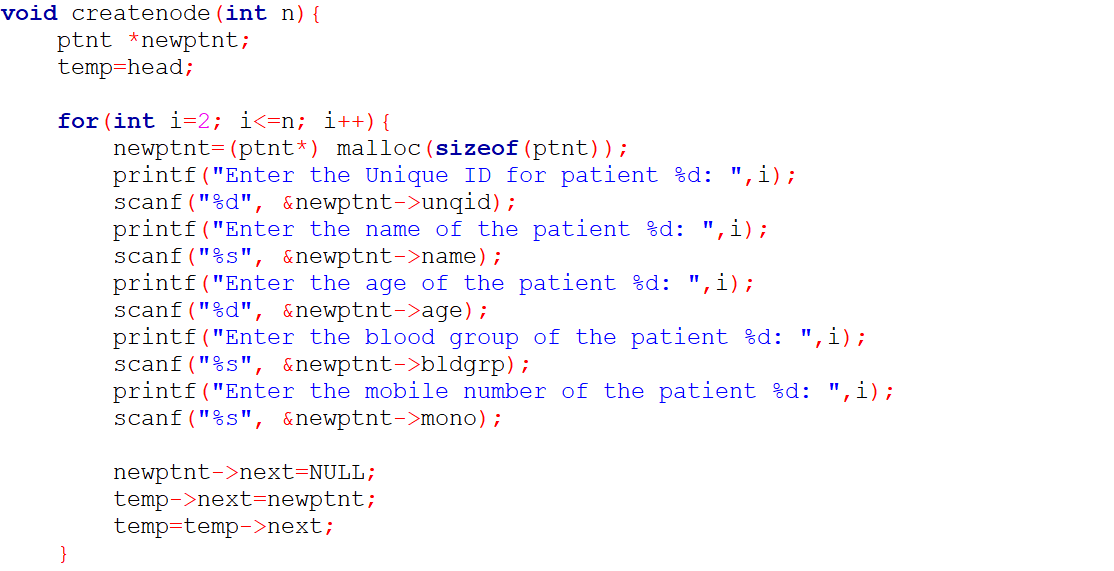
1. We start the code with the “struct” by making a structure which is the fundamental of making a linked list. We take all the data types we need to get the information of patient.



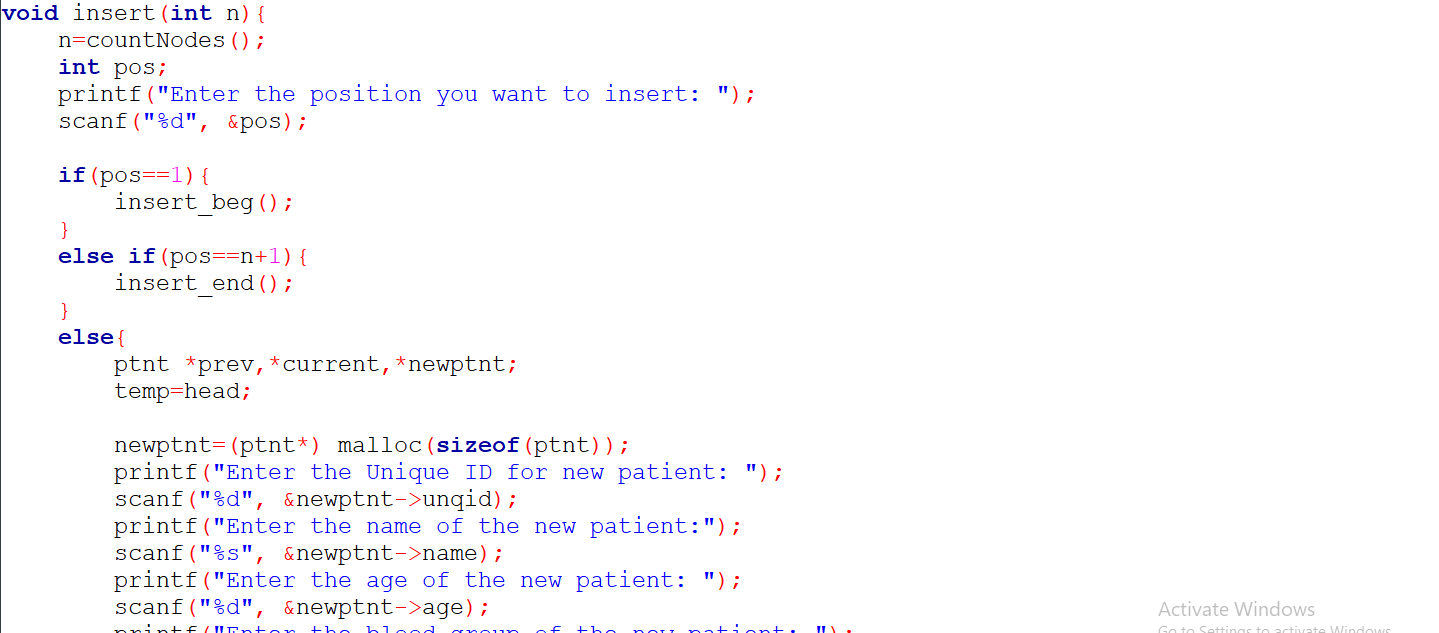
2. After that we start the program by taking commands from the user using “switch-case”. This helps user to use the program continuously as long as he wants by giving command again & again.

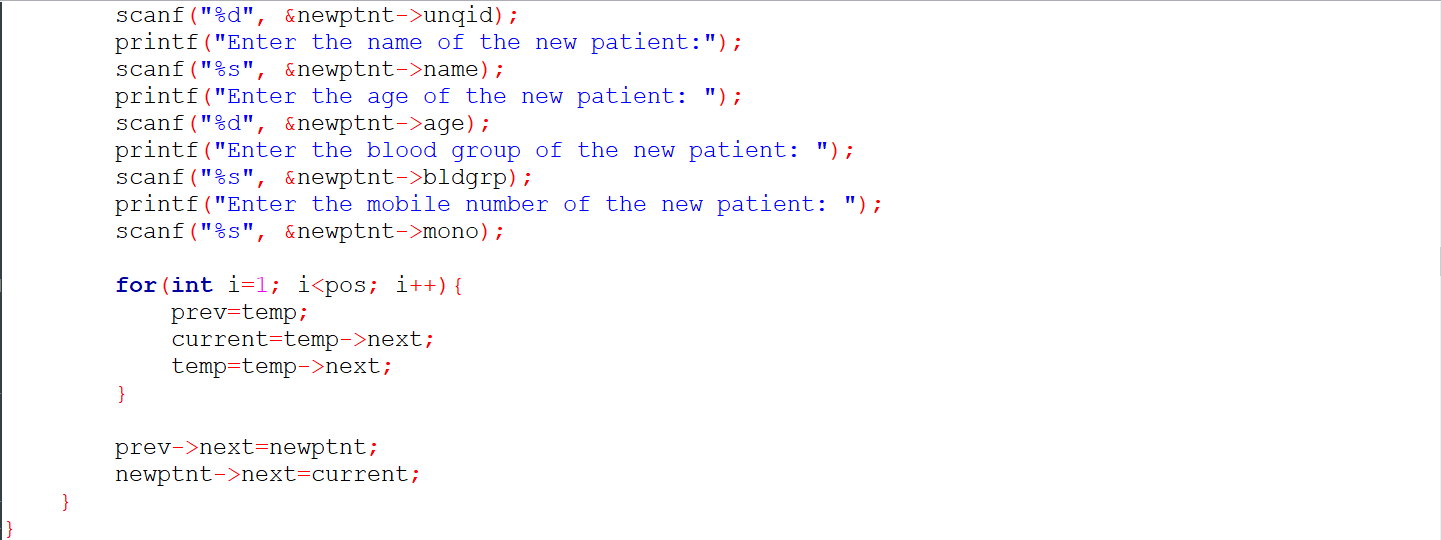


3. Now this code is about the node creation where all the information of a patient will be stored in a node and will be connected to the next node. Here the program will take the Unique ID, Name, Age, Blood Group & Mobile Number of the patients and store the data. The data will be used in further functions.

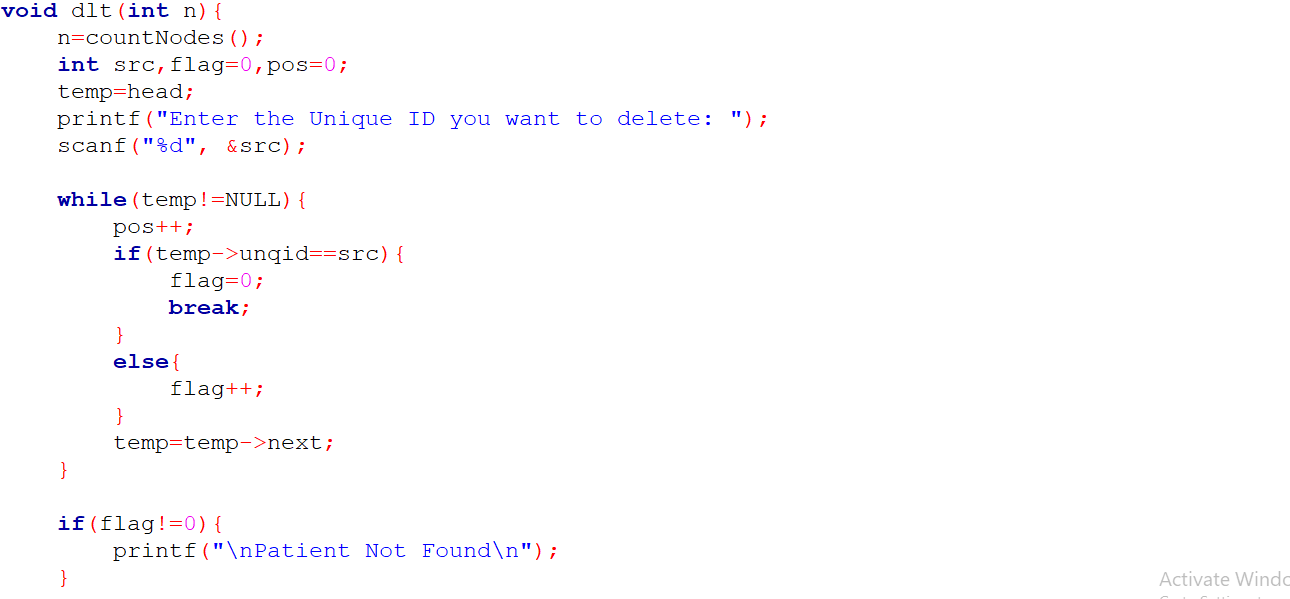


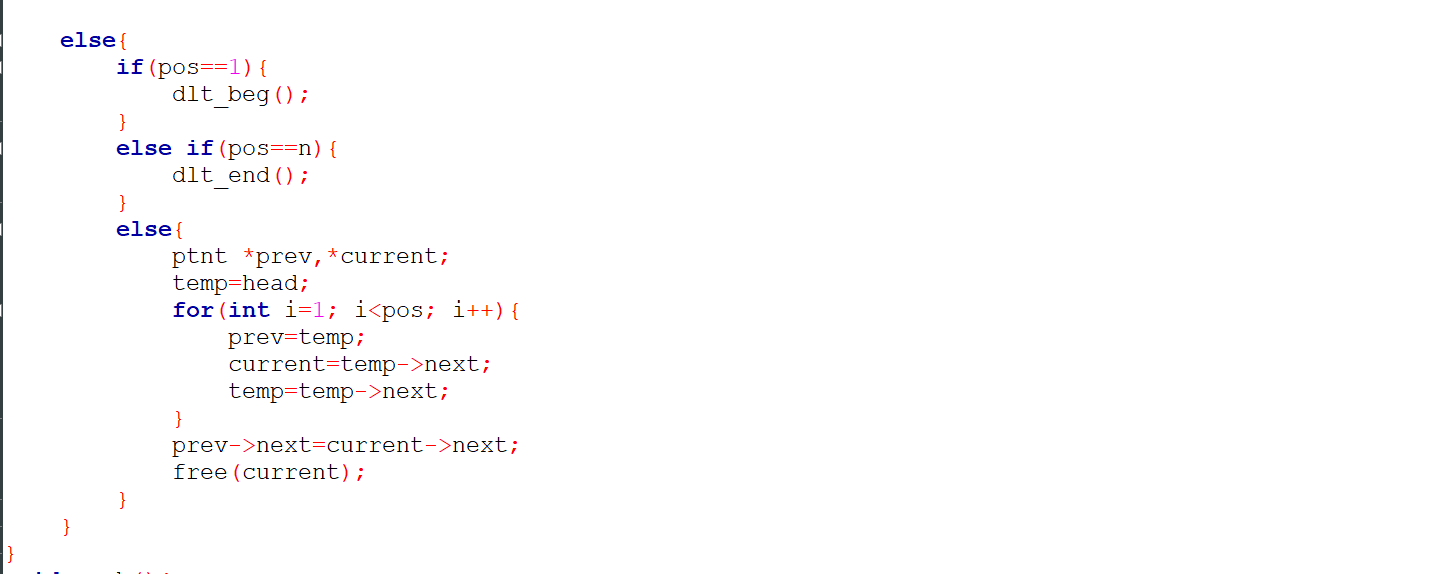
4. After taking the information of patients we may need to insert or delete any of the patient’s information when needed. So we use insert and delete function here to do beginning, middle or end insertion and deletion. Here, we take the position where to insert the data and insert them according to the position by using written functions by us.



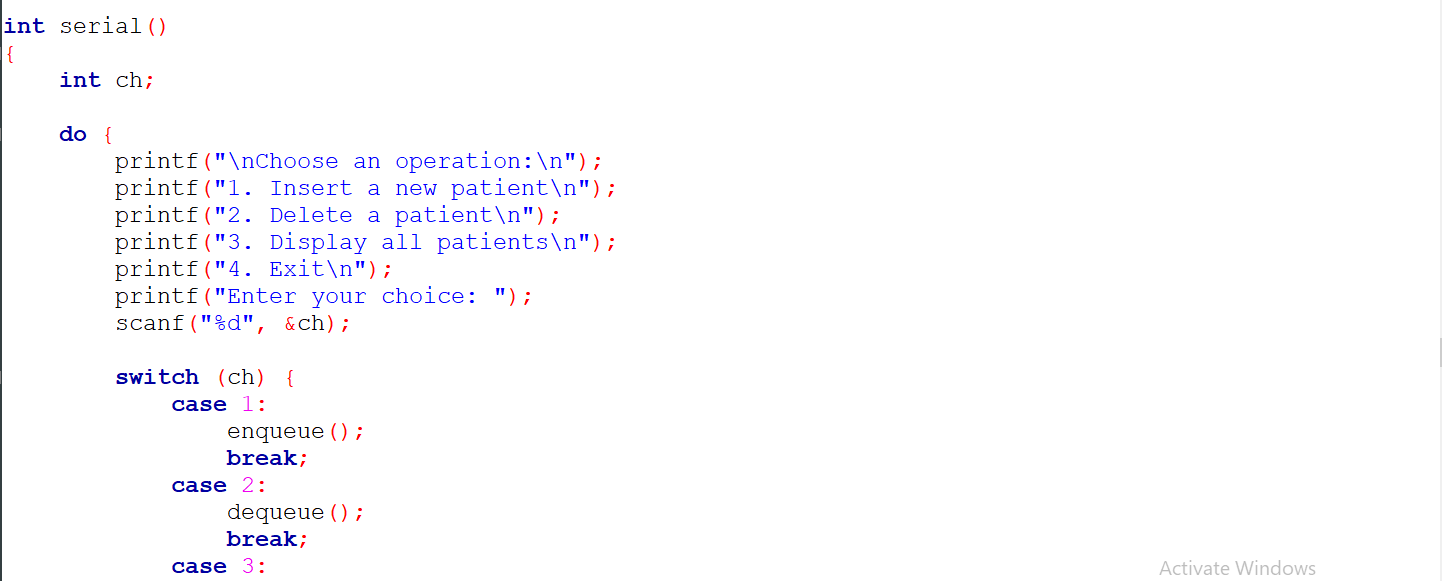
****

5. We don’t delete the patient by taking the position because it is not efficient and the position of the patient can be changed at any time when new patients are added. So we take the unique ID from the patient and search by that ID. If found, then we delete the information by using our written functions.

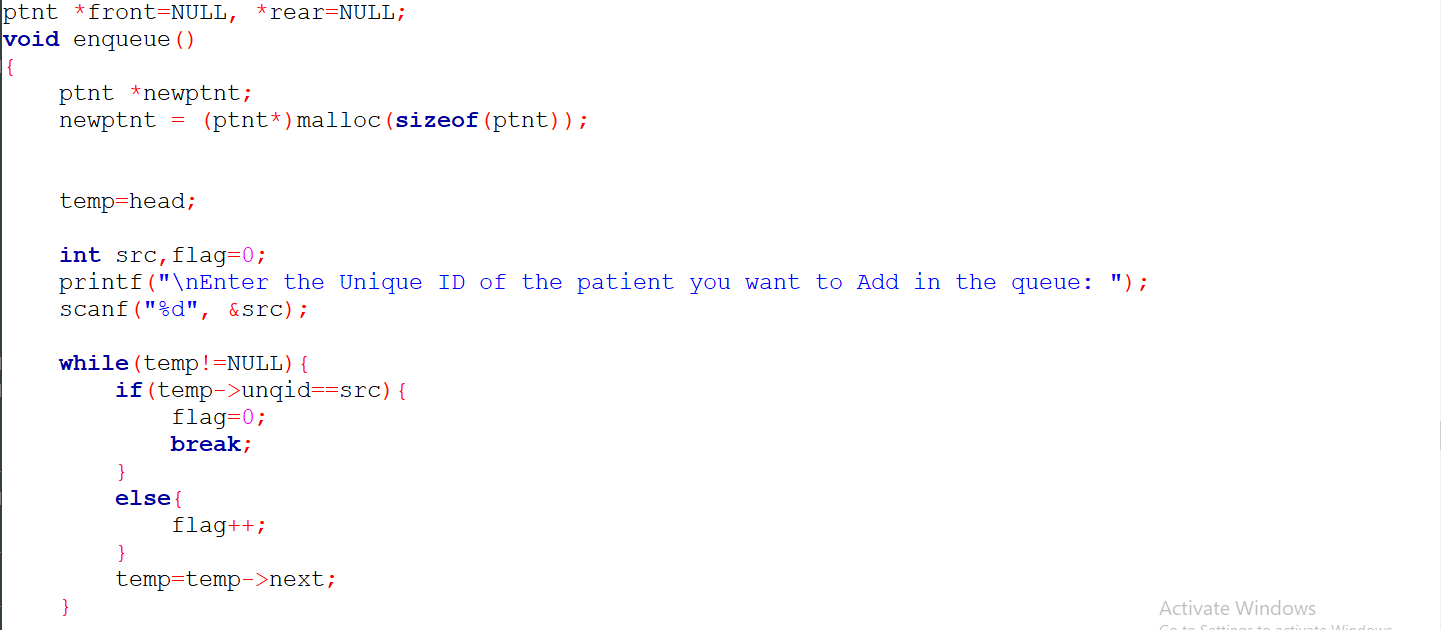


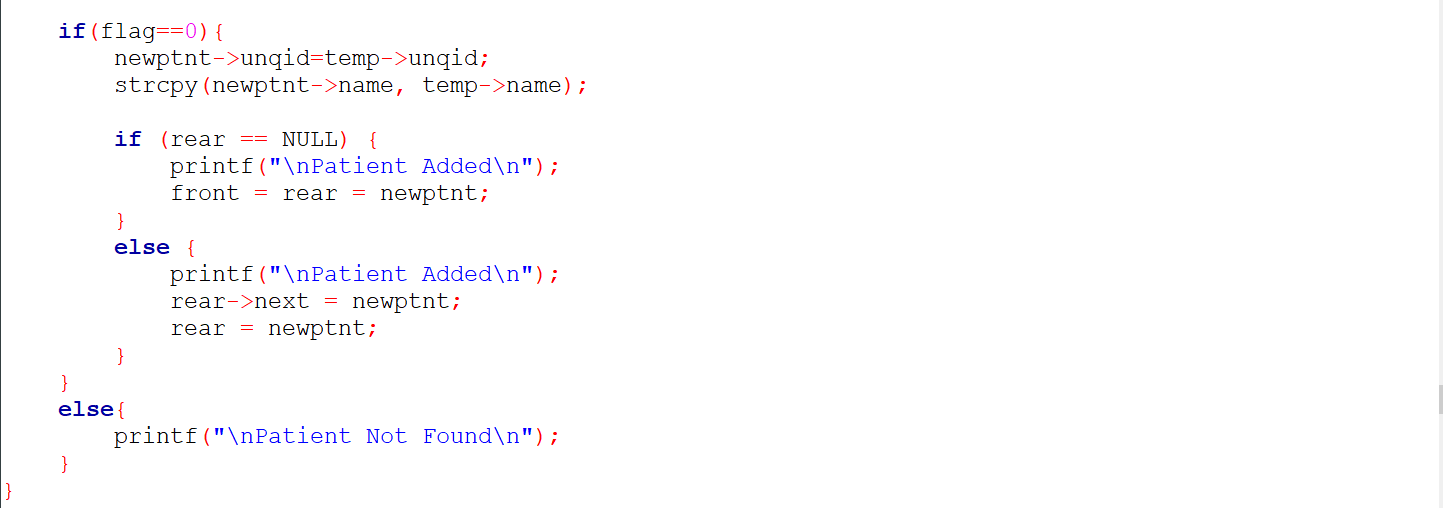


6. After this we have our “Queue Operations”. Here we use switch case to insert, delete and display the data of the patients in serial.



7. We do not allow anyone in the serial without registration, so we take the data of the patients who want to be in serial from the registration earlier. We search for the registered patient using their unique ID and if found we add him to the serial in a queue.

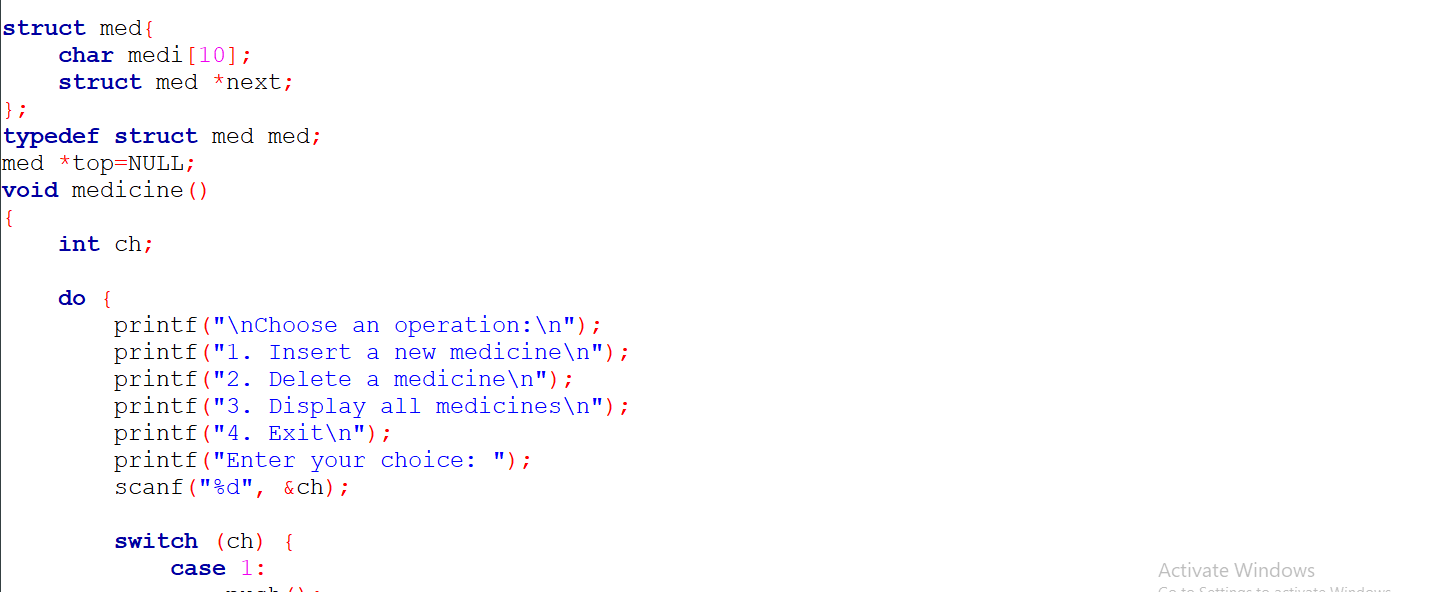




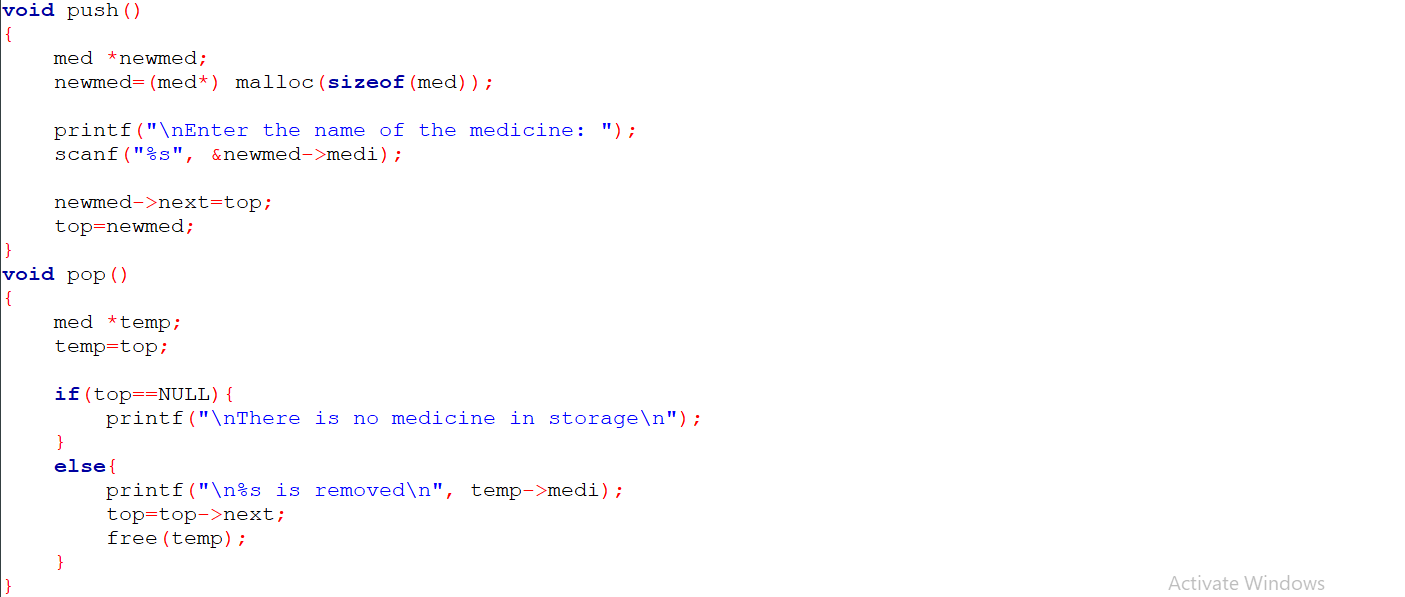
8. We also have to dequeue the patient in serial. As we use First In First Out policy here, so who came first will go first. This rule is followed by queue data structure. So we use dequeue() function here to remove the patient from the serial list when he/she is observed.



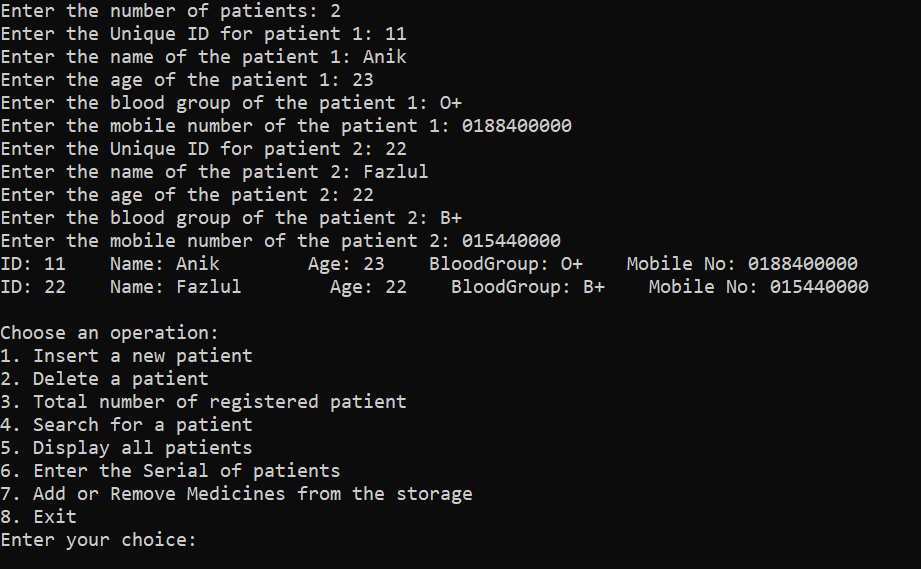
9. After this comes the pharmacy sector where medicines are stored in stack. We also use switch case here to insert, delete or display all the medicines.



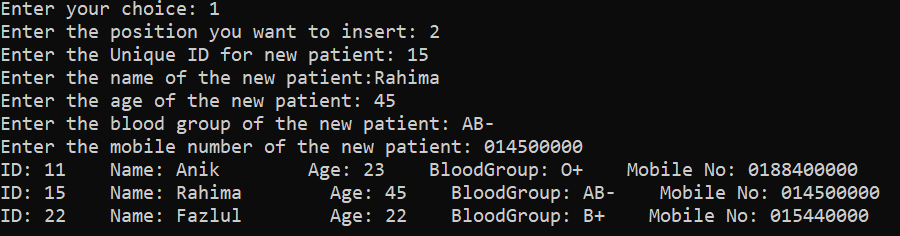
10. As we want to store the newest medicine on the top, we use stack data structure where push() function is used to insert an element in the list and pop() is used to delete it from the top. As a result, patients will always get the latest medicines we have.



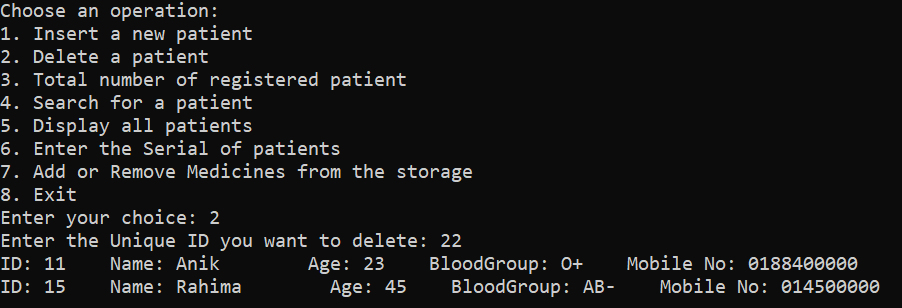
**Output**



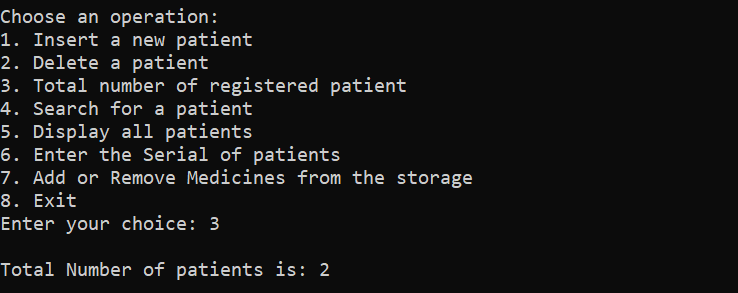
Here the system is taking information from patient and asking instruction for the next operation using switch-case.



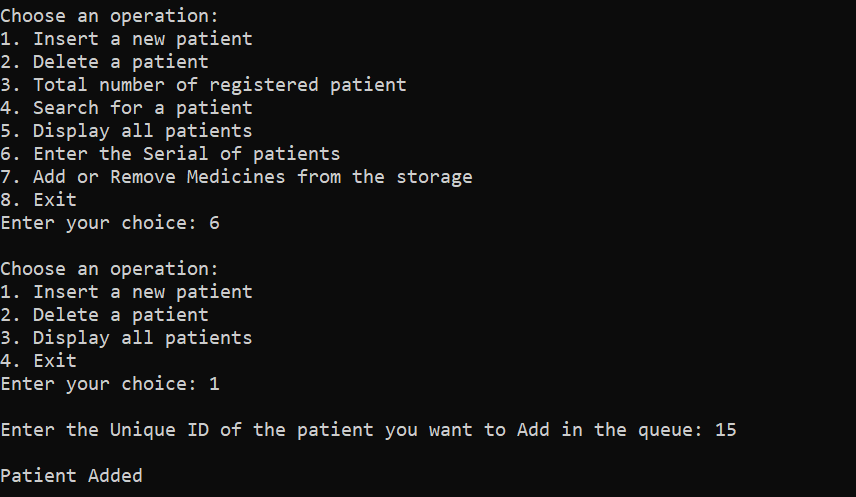
By choosing 1 any new patient can be inserted at any position.



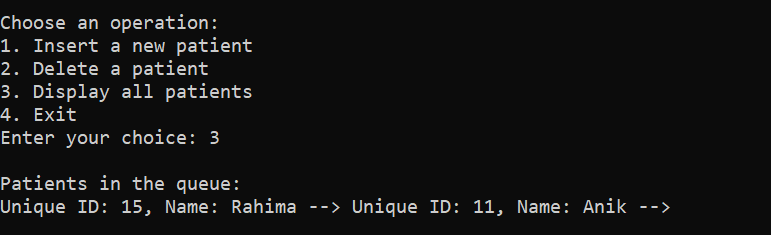
If we choose 2 from the switch case, we can delete any of the patient by searching them by their unique ID.



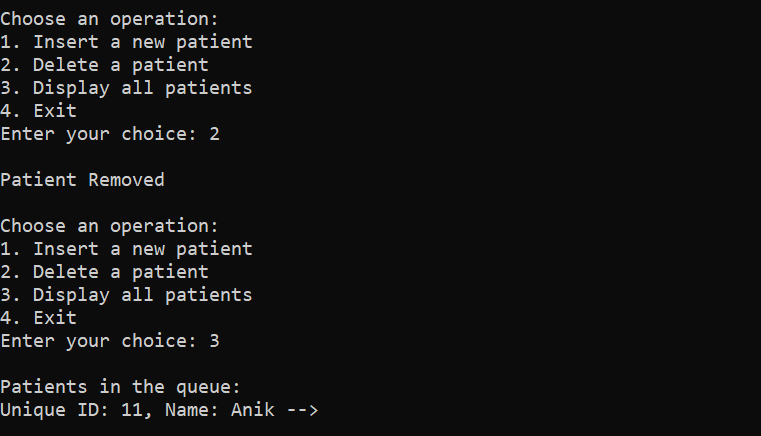
It is also possible to count the total number of patient by choosing 3 from the options.



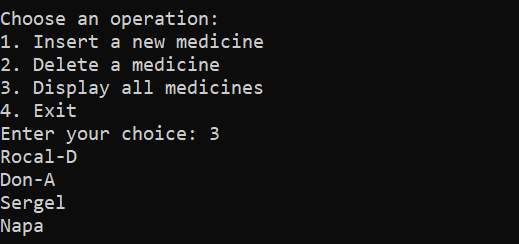
As required, if we want to add any patient in the serial we would have to choose 6 from the options. Then we will get another switch case. This helps to insert or delete any patient from the serial. By choosing 1, we insert any patient by using their registered unique ID.



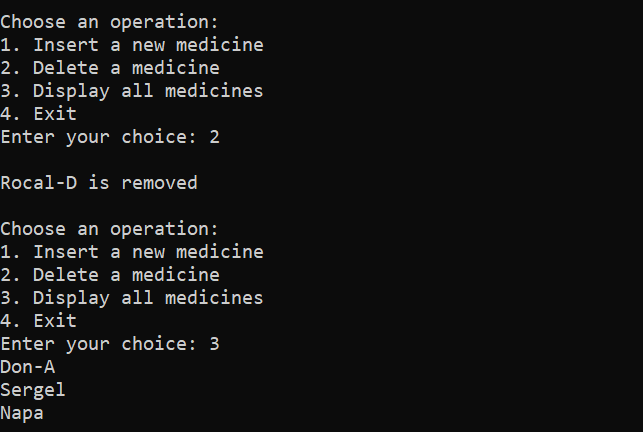
From this operations, we can also see the serial by choosing 3. The system will show us the serial of the patient in order as we used queue data structure here.



After checking one patient the system will have to remove the patient from the serial. It will remove him from the front of the queue as the front one will get the service at first. So, here when we removed a patient we can see the first patient was removed from the queue.



Here comes another section of our system, Pharmacy section. To access this section we have to choose 7 from the main switch case. Here in the section we insert the data of the medicine in stack data structure. We can see that what we inserted here are displayed in a stack.



From a stack, when we remove something we remove it from the top. As expected, the system removed the top medicine here at first when we wanted to dispatch a medicine.

**7. Conclusion**

The Hospital Management System, which was created with C programming and basic data structures, provides an effective solution for managing patient registration, serialization, and medications.

Key accomplishments:

**Patient Registration:** With a unique ID, the system provides for flexible patient entry, search, and deletion.

**Serial Management:** Serialization is organized using a queue data structure, assuring fair treatment based on the "first-come-first-served" principle.

**Pharmacy:** Medication processing is optimized utilizing a stack data structure for the most recent prescription accessibility.

The Hospital Management System improves hospital operations greatly. In summary, this project improves patient management, serialization, and medicine delivery, with future advances potential to change healthcare facility operations.