**Data Structures Mini Project**

**Topic:**

Hostel Management System using **Linked list**, **Queue** and **Files**.

**Members:**

1. Amith Kumar N - 185001018
2. Aravind P - 185001021
3. Bharath Kumar A M R - 185001034

**AIM:**

The aim of this mini project is to make a simple user friendly hostel management system efficiently using the data structures **Linked List** and **Queue** and to perform various basic functions like insert, delete and display effectively.

**Data Structures Used:**

* + Linked list – For managing the room allotments.
  + Queue (linked list implementation) – For managing the waiting list
  + File handling – For managing the database.

**Description:**

* The Program contains **3 types** of hostels (single, shared and international), hence we use **3 linked list** for storing the details of allocated rooms to each hostel and **3 Queues** to maintain the waiting list.
* The database of the hostel management system is managed using Files in c. The structures are written into binary files as objects using fread(). Also text file is used to show the database details.
* Here we can allocate the rooms to the people as per their requests. If the rooms are not vacant the person is pushed into the queue. When an already existing person vacates the room. The waiting list is updated and the person is allotted room.
* The details of the waiting list and existing people in hostel can be written to a text file for easy record maintenances.

**Linked List:**

Like arrays, Linked List is a linear data structure. Unlike arrays, linked list elements are not stored at a contiguous location; the elements are linked using pointers.

[](https://media.geeksforgeeks.org/wp-content/cdn-uploads/gq/2013/03/Linkedlist.png)

**Why Linked List?**

Arrays can be used to store linear data of similar types, but arrays have the following limitations.

**1)** The size of the arrays is fixed: So we must know the upper limit on the number of elements in advance. Also, generally, the allocated memory is equal to the upper limit irrespective of the usage.

**2)** Inserting a new element in an array of elements is expensive because the room has to be created for the new elements and to create room existing elements have to be shifted.

**Advantages over arrays**

**1)** Dynamic size  
**2)** Ease of insertion/deletion

**Drawbacks:**

**1)** Random access is not allowed. We have to access elements sequentially starting from the first node. So we cannot do binary search with linked lists efficiently with its default implementation.

**2)** Extra memory space for a pointer is required with each element of the list.

**3)** Not user friendly. Since array elements are contiguous locations, there is locality of reference which is not there in case of linked lists.

**Queue:**

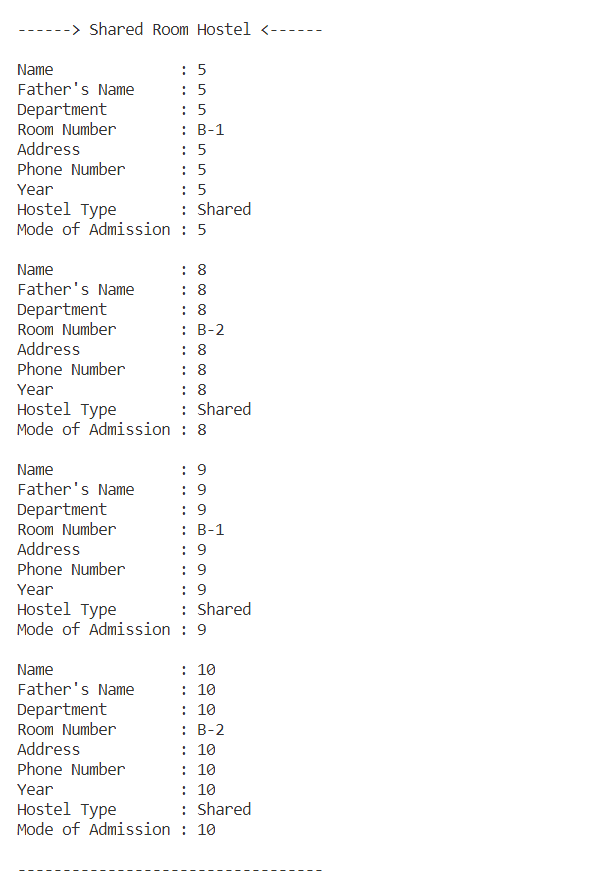
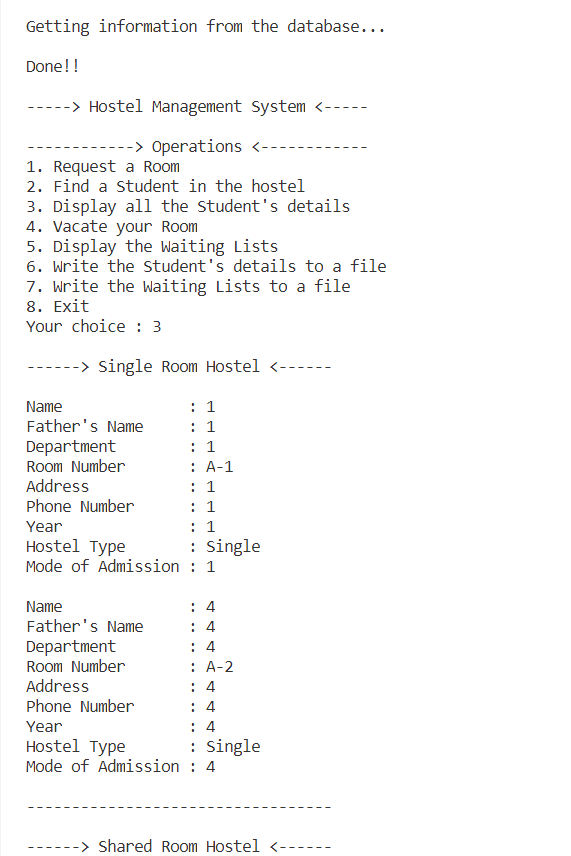
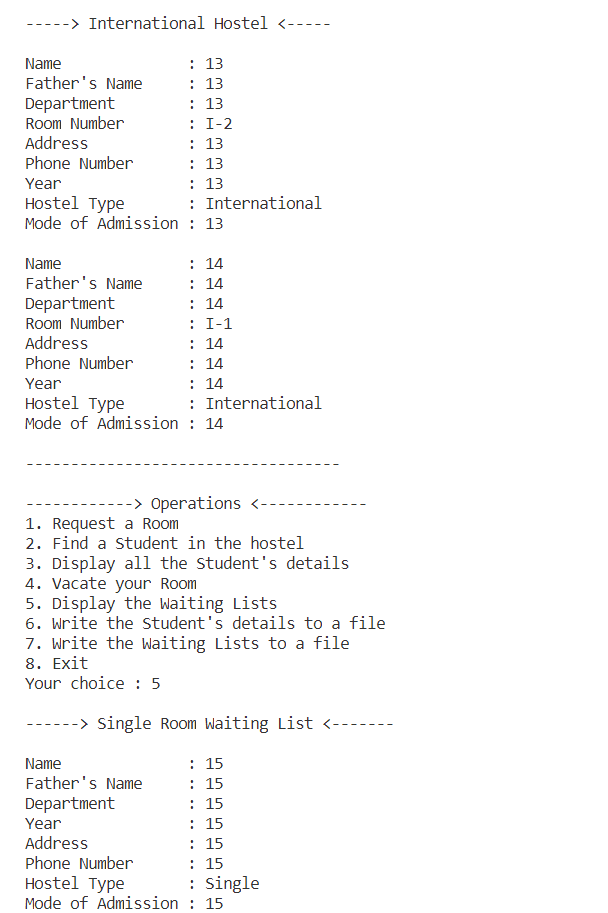
A Queue is a linear structure which follows a particular order in which the operations are performed. The order is First In First Out (FIFO). We maintain two pointers, front and rear. The front points the first item of queue and rear points to last item.

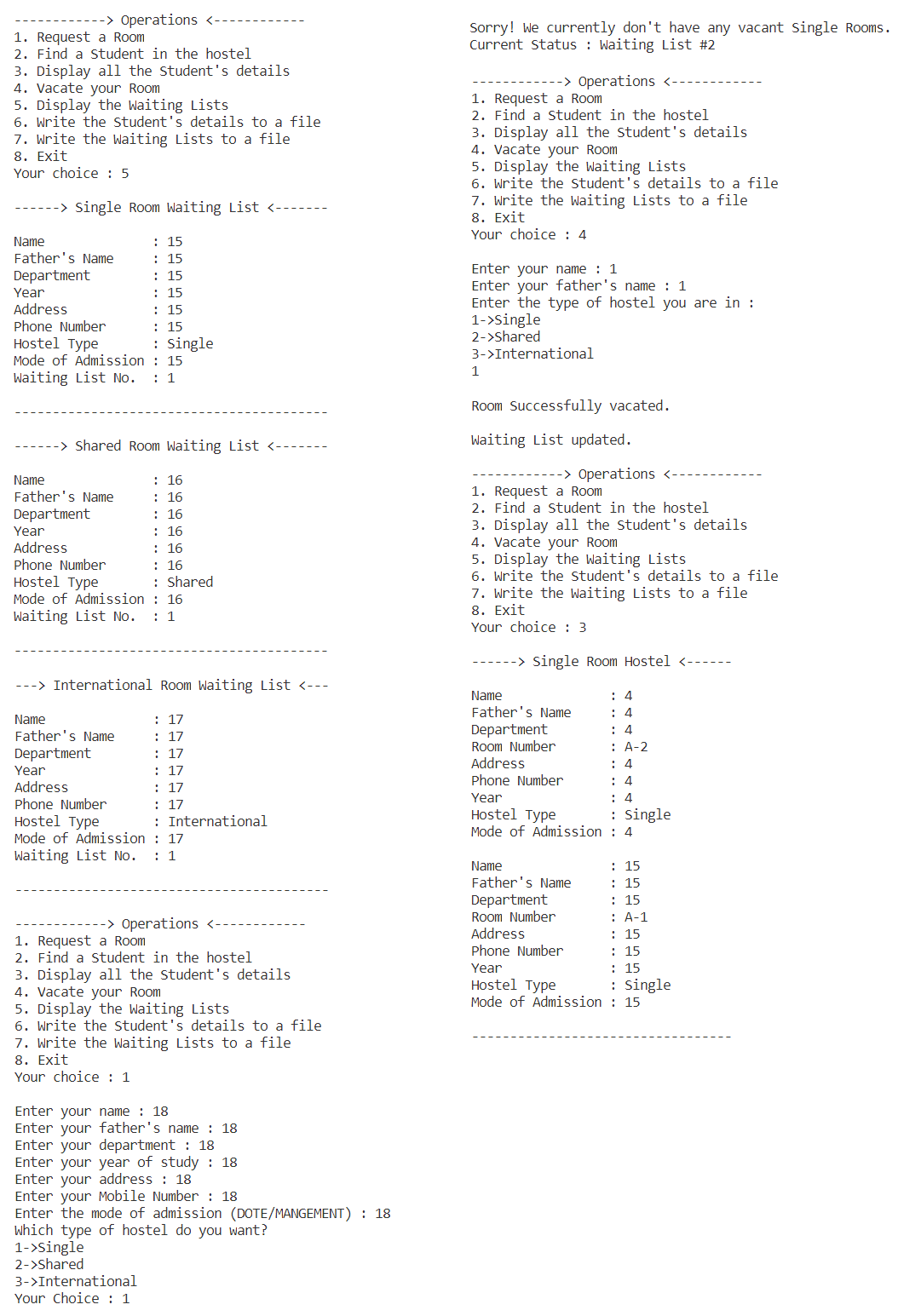


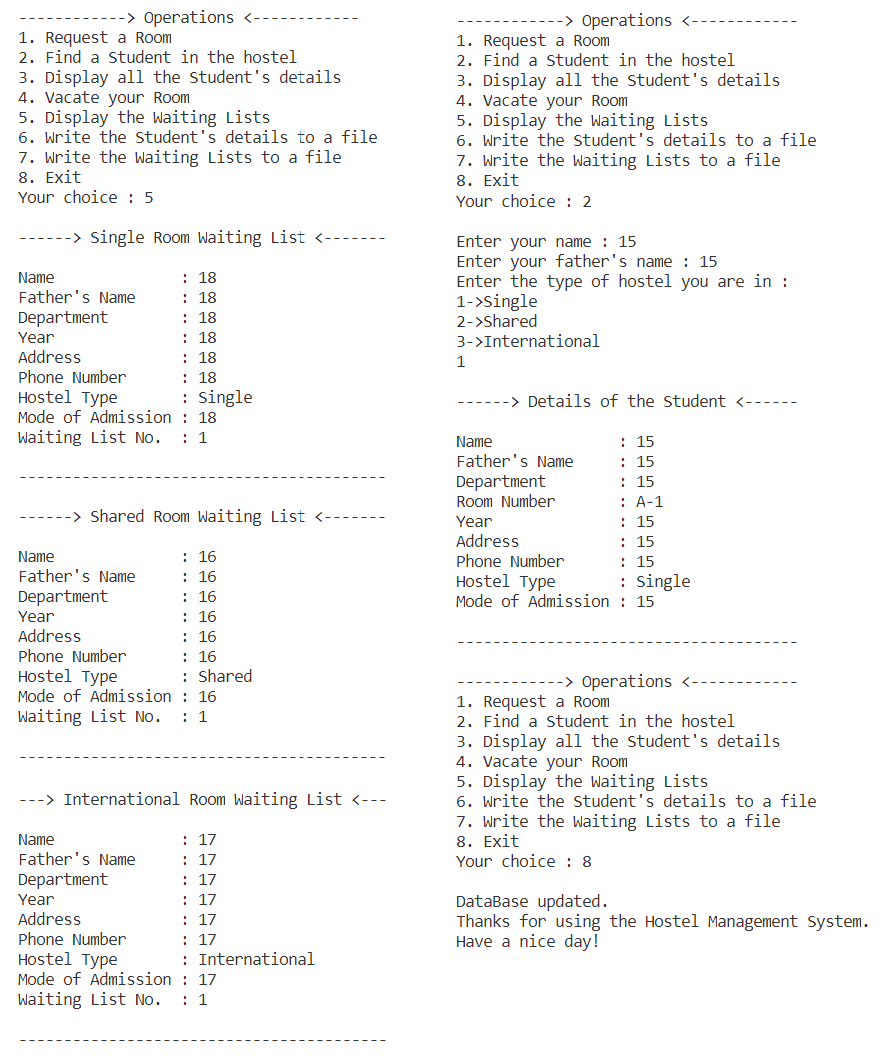
**Learning Experience:**

* We have learnt to organize the data into linked list and queue.
* The Hostel management system contains ADT’s which perform operations like allocating room, display, write to file, vacate the room, manage waiting list, etc.,
* We learnt to manage huge amount to data efficiently by using structures and binary files.
* We have learnt file handling in C to maintain the database for the Hostel Management System.
* The basic linked list and queue operations like insert, delete, find, enqueue, dequeue , etc., are understood.
* The file operations like fread, fwrite, fprintf, fscanf are understood.
* We have learnt to create an ADT with members and objects.
* We have learnt to develop an application using Linked list and Queue.

**Output Screenshots:**

** **

****

****

**Result:**

The concept of data structures is implemented in this application and the Hostel Management System is efficiently developed.