Hospital Patient Management System (Queue,linkedlist,heap,tree)

**Abstract**

The **Hospital Patient Management System** is a software application aimed at enhancing the efficiency of hospital operations by managing patient information, appointments, and medical records systematically. The system leverages advanced data structures to ensure optimized performance for various tasks, including patient registration, emergency prioritization, and hierarchical categorization of patients based on departments. The use of **queues, linked lists, heaps, and trees** allows the system to handle large volumes of data dynamically while ensuring quick and effective access to critical information.

Key Features:

1. **Queue**: Used for handling patient appointments based on the first-come, first-served principle.
2. **Linked List**: Provides a dynamic and flexible way to maintain patient records, accommodating changes without restructuring.
3. **Heap**: Utilized for prioritizing emergency cases, ensuring urgent patients receive timely care.
4. **Tree**: Facilitates the organization of data hierarchically, such as department-wise classification of patients or doctors.

The primary goal of the system is to reduce administrative burden, minimize wait times for patients, and streamline hospital operations to improve overall patient care.

**Detailed Project Report**

**Introduction**

The Hospital Patient Management System addresses the critical need for efficient handling of patient information in hospitals. The system ensures that patient data is stored, retrieved, and managed effectively, reducing manual errors and delays.

**Objectives**

* To provide a user-friendly interface for managing patient data.
* To ensure high efficiency in handling dynamic and large-scale data.
* To prioritize patients based on urgency through systematic categorization.

**Requirements**

1. **Hardware**:
   * Minimum 4GB RAM.
   * 64-bit processor.
   * At least 500MB of storage.
2. **Software**:
   * Operating System: Windows, Linux, or MacOS.
   * Compiler: GCC or any C compiler.

**Design**

The system uses the following data structures:

* **Queue**: For appointment scheduling and managing patient flow.
* **Linked List**: For maintaining patient records that can grow or shrink dynamically.
* **Heap**: For prioritization of critical cases, allowing fast retrieval of the highest-priority patient.
* **Tree**: For organizing data, such as categorizing patients under specific hospital departments.

**Implementation**

Each component of the system is implemented using C language. Below is an example of one of the modules:

**Expanded Project Code Example**

Here’s a more comprehensive code snippet demonstrating linked list and heap integration:

c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Linked List for patient records

struct Patient {

int id;

char name[50];

int priority; // Priority for emergency cases

struct Patient\* next;

};

// Function to add a new patient

void addPatient(struct Patient\*\* head, int id, char\* name, int priority) {

struct Patient\* newPatient = (struct Patient\*)malloc(sizeof(struct Patient));

newPatient->id = id;

strcpy(newPatient->name, name);

newPatient->priority = priority;

newPatient->next = \*head;

\*head = newPatient;

}

// Heap for prioritizing emergency cases

void heapify(int heap[], int n, int i) {

int largest = i;

int left = 2 \* i + 1;

int right = 2 \* i + 2;

if (left < n && heap[left] > heap[largest])

largest = left;

if (right < n && heap[right] > heap[largest])

largest = right;

if (largest != i) {

int temp = heap[i];

heap[i] = heap[largest];

heap[largest] = temp;

heapify(heap, n, largest);

}

}

void insertHeap(int heap[], int\* size, int value) {

heap[\*size] = value;

(\*size)++;

for (int i = (\*size) / 2 - 1; i >= 0; i--)

heapify(heap, \*size, i);

}

// Main function

int main() {

// Linked List Example

struct Patient\* head = NULL;

addPatient(&head, 1, "John Doe", 2);

addPatient(&head, 2, "Jane Smith", 5);

printf("Patient Records:\n");

struct Patient\* temp = head;

while (temp != NULL) {

printf("ID: %d, Name: %s, Priority: %d\n", temp->id, temp->name, temp->priority);

temp = temp->next;

}

// Heap Example

int heap[100];

int size = 0;

insertHeap(heap, &size, 3);

insertHeap(heap, &size, 5);

insertHeap(heap, &size, 1);

printf("\nEmergency Priorities (Heap):\n");

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

printf("%d ", heap[i]);

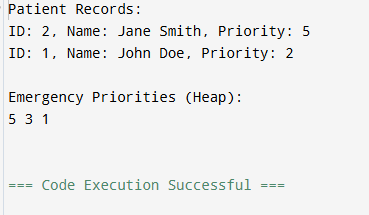
}

printf("\n");

return 0;

}

OUTPUT:



-----------------------\*\*\*\*\*\*\*---------------------------