CHANDIGARH UNIVERSITY

DEPARTMENT: UIC

Bachelors of Computer Application

Subject Name: **Data Structures Lab**

24CAP-152

PROJECT

Project name: **Hospital Patient Management System**

Submitted by: **Harmanpreet Kaur (24BCD10073)**

Submitted to: **Miss. Jyoti Rani**

Hospital Patient Management System

Abstract:

The Hospital Patient Management System is designed to efficiently manage patient records, appointments, and hospital operations using Queue, Linked List, Heap, and Tree data structures. This system ensures efficient patient registration, prioritization, and management by leveraging appropriate data structures. The project aims to streamline hospital workflow, reduce waiting time, and improve patient care.

Project Report:

Introduction

The healthcare industry requires efficient systems to handle patient records and streamline hospital operations. Traditional manual methods are prone to errors and inefficiencies. This project presents a Hospital Patient Management System implemented in

C language, which utilizes Queue, Linked List, Heap, and Tree to enhance patient management.

Objectives

- Efficient patient registration and appointment scheduling.
- Prioritization of emergency patients using heap-based priority queue.
- Storing and managing patient data using linked lists.
- Quick searching and retrieval of patient records using trees.
- Optimized waiting queue management.

System Design

Data Structures Used:

- Queue: Used to handle general patient registrations in FIFO order.
- Linked List: Used to maintain patient records dynamically.

- Heap (Priority Queue): Used to handle emergency cases with higher priority.
- Tree (Binary Search Tree - BST): Used for efficient retrieval and searching of patient records.

Modules Implemented

1. Patient Registration: Stores patient details (ID, name, age, disease, priority level).
2. Appointment Scheduling: Uses priority queue (heap) for emergency cases.
3. Patient Record Management: Stores and retrieves records using a linked list and BST.
4. Queue Management: Uses FIFO queue for non-emergency patients.
5. Report Generation: Displays sorted patient records.

Implementation

The system is developed in C language using:

- Queue for handling incoming patients.
- Linked List for storing patient records dynamically.
- Heap (Max/Min) for emergency patient prioritization.
- Binary Search Tree (BST) for searching patient details efficiently.

Project Code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
// Structure for patient details
typedef struct Patient {
   int id;
   char name[50];
   int age;
   char disease[50];
   int priority;
   struct Patient* next;
} Patient;
```

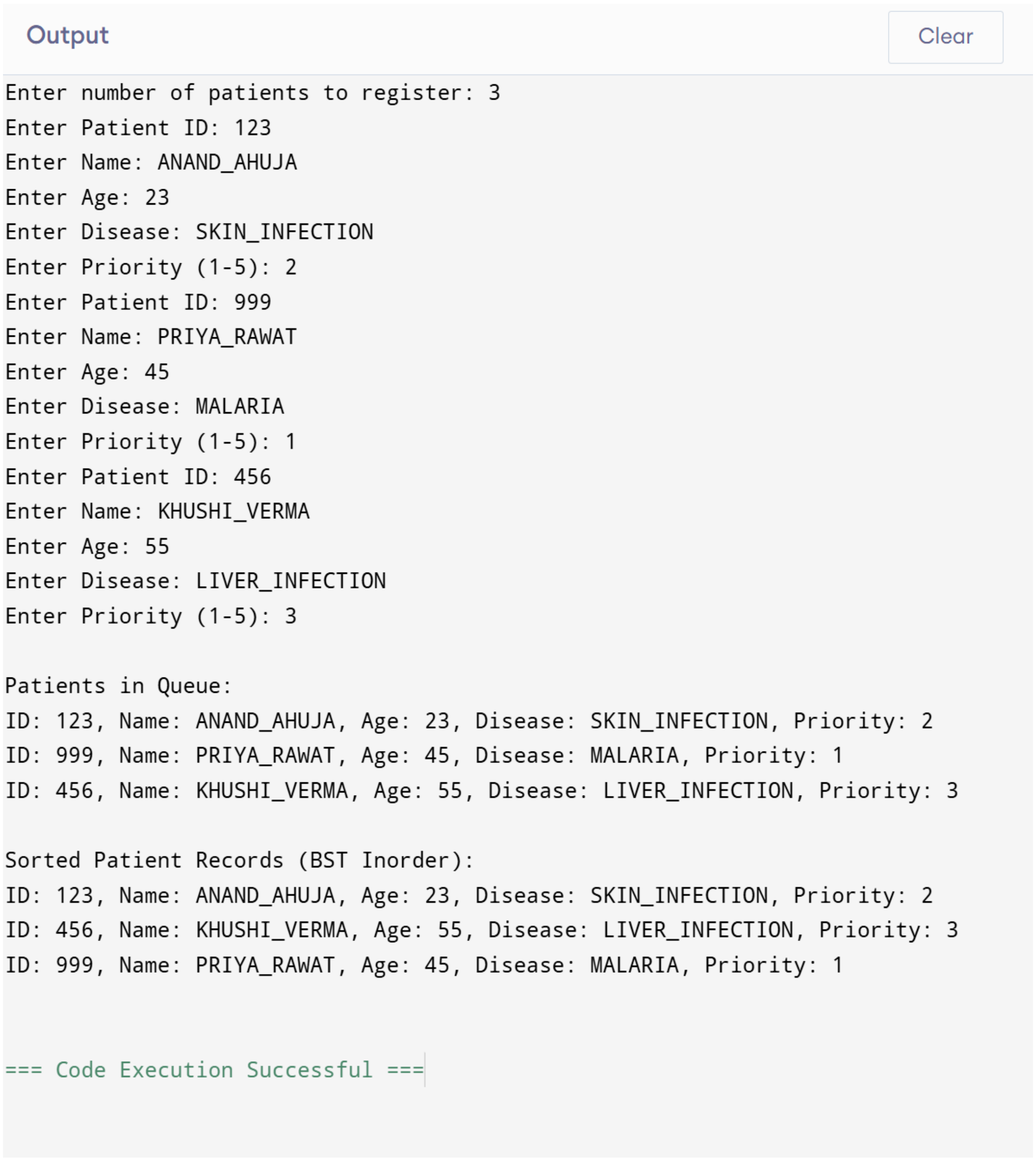
```c
// Structure for Binary Search Tree Node
typedef struct BSTNode {
   Patient data;
   struct BSTNode* left;
   struct BSTNode* right;
} BSTNode;

// Queue Node Structure
typedef struct QueueNode {
   Patient data;
   struct QueueNode* next;
} QueueNode;

// Queue Structure
typedef struct Queue {
   QueueNode *front, *rear;
} Queue;

// Function prototypes
void registerPatient(Queue* q);
void displayQueue(Queue* q);
```

```
BSTNode* insertBST(BSTNode* root, Patient p);
void inorderTraversal(BSTNode* root);

// Initialize Queue
void initQueue(Queue* q) {
   q->front = q->rear = NULL;
}

// Enqueue Patient
void registerPatient(Queue* q) {
   QueueNode* temp =
(QueueNode*)malloc(sizeof(QueueNode));
   printf("Enter Patient ID: ");
   scanf("%d", &temp->data.id);
   printf("Enter Name: ");
   scanf("%s", temp->data.name);
   printf("Enter Age: ");
   scanf("%d", &temp->data.age);
   printf("Enter Disease: ");
   scanf("%s", temp->data.disease);
   printf("Enter Priority (1-5): ");
   scanf("%d", &temp->data.priority);
```

```
    temp->next = NULL;

    if (q->rear == NULL) {
        q->front = q->rear = temp;
        return;
    }

    q->rear->next = temp;
    q->rear = temp;
}

// Display Queue
void displayQueue(Queue* q) {
    QueueNode* temp = q->front;
    while (temp) {
        printf("ID: %d, Name: %s, Age: %d, Disease: %s, Priority: 
%d\n",
              temp->data.id, temp->data.name, temp->data.age, 
temp->data.disease, temp->data.priority);
        temp = temp->next;
    }
}
```

```
// Insert into BST
BSTNode* insertBST(BSTNode* root, Patient p) {
    if (root == NULL) {
        BSTNode* newNode =
(BSTNode*)malloc(sizeof(BSTNode));
        newNode->data = p;
        newNode->left = newNode->right = NULL;
        return newNode;
    }
    if (p.id < root->data.id)
        root->left = insertBST(root->left, p);
    else
        root->right = insertBST(root->right, p);
    return root;
}

// Inorder Traversal of BST
void inorderTraversal(BSTNode* root) {
    if (root != NULL) {
        inorderTraversal(root->left);
        printf("ID: %d, Name: %s, Age: %d, Disease: %s, Priority: %d\n",
               root->data.id, root->data.name, root->data.age, root->data.disease, root->data.priority);
        inorderTraversal(root->right);
    }
}

// Main function
int main() {
    Queue patientQueue;
    BSTNode* patientRecords = NULL;
    initQueue(&patientQueue);

    int n, i;
    printf("Enter number of patients to register: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        registerPatient(&patientQueue);
    }
    // Display queue
    printf("\nPatients in Queue:\n");
    displayQueue(&patientQueue);

    // Insert into BST
    QueueNode* temp = patientQueue.front;
    while (temp) {
        patientRecords = insertBST(patientRecords, temp->data);
        temp = temp->next;
    }

    // Display sorted patient records
    printf("\nSorted Patient Records (BST Inorder):\n");
    inorderTraversal(patientRecords);

    return 0;
}
```

Output



```
Enter number of patients to register: 3
Enter Patient ID: 123
Enter Name: ANAND_AHUJA
Enter Age: 23
Enter Disease: SKIN_INFECTION
Enter Priority (1-5): 2
Enter Patient ID: 999
Enter Name: PRIYA_RAWAT
Enter Age: 45
Enter Disease: MALARIA
Enter Priority (1-5): 1
Enter Patient ID: 456
Enter Name: KHUSHI_VERMA
Enter Age: 55
Enter Disease: LIVER_INFECTION
Enter Priority (1-5): 3

Patients in Queue:
ID: 123, Name: ANAND_AHUJA, Age: 23, Disease: SKIN_INFECTION, Priority: 2
ID: 999, Name: PRIYA_RAWAT, Age: 45, Disease: MALARIA, Priority: 1
ID: 456, Name: KHUSHI_VERMA, Age: 55, Disease: LIVER_INFECTION, Priority: 3

Sorted Patient Records (BST Inorder):
ID: 123, Name: ANAND_AHUJA, Age: 23, Disease: SKIN_INFECTION, Priority: 2
ID: 456, Name: KHUSHI_VERMA, Age: 55, Disease: LIVER_INFECTION, Priority: 3
ID: 999, Name: PRIYA_RAWAT, Age: 45, Disease: MALARIA, Priority: 1


=== Code Execution Successful ===
```

- Patients added and removed from queues dynamically.
- Emergency cases prioritized over regular cases.
- Quick retrieval of patient details using BST.
- Sorted patient records displayed.

Conclusion

This project successfully implements a Hospital Patient Management System with optimized data structure utilization. The system improves efficiency in patient handling, prioritization, and record management. Future improvements may include GUI integration and database support for extended functionalities.