#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_NAME\_LEN 50

#define MAX\_CITY\_LEN 30

// Structure for Student

typedef struct Student {

char name[MAX\_NAME\_LEN];

int age;

float mhcet\_percentile;

float jee\_main\_percentile;

char city[MAX\_CITY\_LEN];

struct Student\* next;

} Student;

// Structure for a search request in the queue

typedef struct SearchRequest {

char type; // 'N' for name, 'A' for age, 'P' for percentile

char name[MAX\_NAME\_LEN];

int age;

float percentile;

char exam[10]; // "MHCET" or "JEE"

struct SearchRequest\* next;

} SearchRequest;

// Queue to manage search requests

typedef struct Queue {

SearchRequest\* front;

SearchRequest\* rear;

} Queue;

// Function prototypes

Student\* createStudent(const char\* name, int age, float mhcet, float jee, const char\* city);

void insertStudent(Student\*\* head, Student\* new\_student);

void displayStudent(Student\* student);

void displayStudents(Student\* head);

void searchByName(Student\* head, const char\* name);

void searchByAge(Student\* head, int age);

void searchByPercentile(Student\* head, float percentile, const char\* exam);

Queue\* createQueue();

void enqueue(Queue\* q, char type, const char\* name, int age, float percentile, const char\* exam);

SearchRequest\* dequeue(Queue\* q);

void processQueue(Queue\* q, Student\* head);

void addStudent(Student\*\* head);

int main() {

Student\* head = NULL;

Queue\* searchQueue = createQueue();

// Sample student data

insertStudent(&head, createStudent("Pranav", 18, 92.5, 85.2, "Thane"));

insertStudent(&head, createStudent("Ravi", 19, 88.0, 78.5, "Ulhasnagar"));

insertStudent(&head, createStudent("Omkar", 20, 80.0, 90.1, "Chembur"));

int choice, age;

char name[MAX\_NAME\_LEN], exam[10];

float percentile;

while (1) {

printf("\nUniversity Admission Manager:\n");

printf("1. Display All Students\n");

printf("2. Add New Student\n");

printf("3. Search by Name\n");

printf("4. Search by Age\n");

printf("5. Search by Percentile\n");

printf("6. Process Search Requests\n");

printf("7. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

displayStudents(head);

break;

case 2:

addStudent(&head);

break;

case 3:

printf("Enter name: ");

scanf("%s", name);

enqueue(searchQueue, 'N', name, 0, 0, "");

break;

case 4:

printf("Enter age: ");

scanf("%d", &age);

enqueue(searchQueue, 'A', "", age, 0, "");

break;

case 5:

printf("Enter exam (MHCET/JEE): ");

scanf("%s", exam);

printf("Enter percentile: ");

scanf("%f", &percentile);

enqueue(searchQueue, 'P', "", 0, percentile, exam);

break;

case 6:

processQueue(searchQueue, head);

break;

case 7:

printf("Exiting program...\n");

exit(0);

default:

printf("Invalid choice!\n");

}

}

return 0;

}

// Function to create a new student

Student\* createStudent(const char\* name, int age, float mhcet, float jee, const char\* city) {

Student\* new\_student = (Student\*)malloc(sizeof(Student));

strcpy(new\_student->name, name);

new\_student->age = age;

new\_student->mhcet\_percentile = mhcet;

new\_student->jee\_main\_percentile = jee;

strcpy(new\_student->city, city);

new\_student->next = NULL;

return new\_student;

}

// Function to insert a student into the list

void insertStudent(Student\*\* head, Student\* new\_student) {

new\_student->next = \*head;

\*head = new\_student;

}

// Function to display a single student's data

void displayStudent(Student\* student) {

printf("Name: %s, Age: %d, MHCET: %.2f, JEE: %.2f, City: %s\n",

student->name, student->age, student->mhcet\_percentile, student->jee\_main\_percentile, student->city);

}

// Function to display all students

void displayStudents(Student\* head) {

Student\* current = head;

while (current != NULL) {

displayStudent(current);

current = current->next;

}

}

// Function to add a new student by taking input from the user

void addStudent(Student\*\* head) {

char name[MAX\_NAME\_LEN];

int age;

float mhcet, jee;

char city[MAX\_CITY\_LEN];

printf("Enter name: ");

scanf("%s", name);

printf("Enter age: ");

scanf("%d", &age);

printf("Enter MHCET percentile: ");

scanf("%f", &mhcet);

printf("Enter JEE Main percentile: ");

scanf("%f", &jee);

printf("Enter city: ");

scanf("%s", city);

Student\* new\_student = createStudent(name, age, mhcet, jee, city);

insertStudent(head, new\_student);

printf("Student added successfully.\n");

}

// Search by name

void searchByName(Student\* head, const char\* name) {

Student\* current = head;

int found = 0;

while (current != NULL) {

if (strcmp(current->name, name) == 0) {

displayStudent(current);

found = 1;

}

current = current->next;

}

if (!found) printf("No student found with name %s.\n", name);

}

// Search by age

void searchByAge(Student\* head, int age) {

Student\* current = head;

int found = 0;

while (current != NULL) {

if (current->age == age) {

displayStudent(current);

found = 1;

}

current = current->next;

}

if (!found) printf("No student found with age %d.\n", age);

}

// Search by percentile

void searchByPercentile(Student\* head, float percentile, const char\* exam) {

Student\* current = head;

int found = 0;

while (current != NULL) {

if ((strcmp(exam, "MHCET") == 0 && current->mhcet\_percentile >= percentile) ||

(strcmp(exam, "JEE") == 0 && current->jee\_main\_percentile >= percentile)) {

displayStudent(current);

found = 1;

}

current = current->next;

}

if (!found) printf("No student found with %s percentile >= %.2f.\n", exam, percentile);

}

// Create a queue

Queue\* createQueue() {

Queue\* q = (Queue\*)malloc(sizeof(Queue));

q->front = q->rear = NULL;

return q;

}

// Enqueue a search request

void enqueue(Queue\* q, char type, const char\* name, int age, float percentile, const char\* exam) {

SearchRequest\* new\_request = (SearchRequest\*)malloc(sizeof(SearchRequest));

new\_request->type = type;

strcpy(new\_request->name, name);

new\_request->age = age;

new\_request->percentile = percentile;

strcpy(new\_request->exam, exam);

new\_request->next = NULL;

if (q->rear == NULL) {

q->front = q->rear = new\_request;

} else {

q->rear->next = new\_request;

q->rear = new\_request;

}

}

// Dequeue a search request

SearchRequest\* dequeue(Queue\* q) {

if (q->front == NULL) return NULL;

SearchRequest\* temp = q->front;

q->front = q->front->next;

if (q->front == NULL) q->rear = NULL;

return temp;

}

// Process all search requests in the queue

void processQueue(Queue\* q, Student\* head) {

SearchRequest\* request;

while ((request = dequeue(q)) != NULL) {

printf("\nProcessing search request:\n");

if (request->type == 'N') {

printf("Searching by Name: %s\n", request->name);

searchByName(head, request->name);

} else if (request->type == 'A') {

printf("Searching by Age: %d\n", request->age);

searchByAge(head, request->age);

} else if (request->type == 'P') {

printf("Searching by Percentile in %s: %.2f\n", request->exam, request->percentile);

searchByPercentile(head, request->percentile, request->exam);

}

free(request);

}

}