#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

#define MAX\_QUESTIONS 10

#define QUESTIONS\_TO\_ASK 5

typedef struct {

char question[256];

char options[4][100];

char correct;

} Question;

typedef struct AnswerNode {

int questionNumber;

char playerAnswer;

struct AnswerNode \*next;

} AnswerNode;

// Function Prototypes

void playLevel(Question[], const char \*, AnswerNode \*\*);

void insertAnswer(AnswerNode \*\*, int, char);

void displayAnswers(AnswerNode \*);

int main() {

srand(time(NULL)); // Seed random number generator

// Easy Level Questions

Question easy[MAX\_QUESTIONS] = {

{"What data structure uses FIFO (First In First Out)?", {"a) Stack", "b) Queue", "c) Tree", "d) Graph"}, 'b'},

{"Which data structure uses LIFO (Last In First Out)?", {"a) Queue", "b) Stack", "c) Linked List", "d) Tree"}, 'b'},

{"An array index starts from?", {"a) 0", "b) 1", "c) 2", "d) -1"}, 'a'},

{"Which of these is a linear data structure?", {"a) Tree", "b) Graph", "c) Stack", "d) Heap"}, 'c'},

{"Which data structure is used for recursion?", {"a) Queue", "b) Stack", "c) Array", "d) Linked List"}, 'b'},

{"What type of data structure is an array?", {"a) Dynamic", "b) Static", "c) None", "d) Both"}, 'b'},

{"What is the maximum number of children a binary tree node can have?", {"a) 1", "b) 2", "c) 3", "d) 4"}, 'b'},

{"Which structure allows insertion and deletion from both ends?", {"a) Stack", "b) Deque", "c) Queue", "d) Array"}, 'b'},

{"Best case time complexity for searching in a sorted array?", {"a) O(1)", "b) O(n)", "c) O(log n)", "d) O(n log n)"}, 'a'},

{"What is NULL pointer used for in linked lists?", {"a) Start", "b) End", "c) Middle", "d) None"}, 'b'}

};

// Medium Level Questions

Question medium[MAX\_QUESTIONS] = {

{"In a linked list, insertion at the beginning is?", {"a) O(1)", "b) O(log n)", "c) O(n)", "d) O(n log n)"}, 'a'},

{"Which traversal is Left-Root-Right in a binary tree?", {"a) Preorder", "b) Inorder", "c) Postorder", "d) Levelorder"}, 'b'},

{"Which sorting algorithm is best for nearly sorted arrays?", {"a) Bubble Sort", "b) Insertion Sort", "c) Selection Sort", "d) Quick Sort"}, 'b'},

{"What is the height of an empty tree?", {"a) -1", "b) 0", "c) 1", "d) 2"}, 'a'},

{"Which heap is used in a Priority Queue?", {"a) Min Heap", "b) Max Heap", "c) Binary Search Tree", "d) AVL Tree"}, 'a'},

{"What is the time complexity of binary search?", {"a) O(1)", "b) O(n)", "c) O(log n)", "d) O(n log n)"}, 'c'},

{"A graph with no cycles is called?", {"a) Tree", "b) Directed Graph", "c) Undirected Graph", "d) Complete Graph"}, 'a'},

{"In which case is bubble sort’s performance best?", {"a) Random elements", "b) Already sorted", "c) Reverse order", "d) Partially sorted"}, 'b'},

{"Stack is mainly used for?", {"a) BFS", "b) DFS", "c) Both", "d) Neither"}, 'b'},

{"Queue is mainly used for?", {"a) DFS", "b) BFS", "c) Inorder traversal", "d) Preorder traversal"}, 'b'}

};

// Hard Level Questions

Question hard[MAX\_QUESTIONS] = {

{"Which data structure is used in Dijkstra’s algorithm?", {"a) Stack", "b) Queue", "c) Priority Queue", "d) Array"}, 'c'},

{"Which is not a self-balancing tree?", {"a) AVL", "b) Red-Black", "c) B-Tree", "d) Binary Search Tree"}, 'd'},

{"What is the degree of a node?", {"a) Depth", "b) Number of children", "c) Height", "d) Number of edges"}, 'b'},

{"Which traversal of BST gives ascending order?", {"a) Inorder", "b) Preorder", "c) Postorder", "d) Level order"}, 'a'},

{"In hashing, what is a collision?", {"a) No space left", "b) Two keys hash to same value", "c) Key not found", "d) Search failure"}, 'b'},

{"Minimum number of edges in a tree with n nodes?", {"a) n", "b) n+1", "c) n-1", "d) n/2"}, 'c'},

{"Which of the following is not O(log n)?", {"a) Binary Search", "b) AVL Insertion", "c) Heap Insertion", "d) Linear Search"}, 'd'},

{"What is the maximum number of nodes at level 'l' in a binary tree?", {"a) 2^l", "b) 2^(l-1)", "c) l", "d) l^2"}, 'a'},

{"What is the average case time complexity of quicksort?", {"a) O(n)", "b) O(n^2)", "c) O(log n)", "d) O(n log n)"}, 'd'},

{"In graph theory, what is a complete graph?", {"a) Each node connected to every other node", "b) No cycles", "c) No edges", "d) Only one edge"}, 'a'}

};

AnswerNode \*answersHead = NULL;

printf("Welcome to the Data Structures Quiz Game!\n\nStarting Easy Level:\n");

playLevel(easy, "Easy", &answersHead);

printf("\nMoving to Medium Level:\n");

playLevel(medium, "Medium", &answersHead);

printf("\nMoving to Hard Level:\n");

playLevel(hard, "Hard", &answersHead);

printf("\nYour Answers:\n");

displayAnswers(answersHead);

// Free linked list memory

AnswerNode \*temp;

while (answersHead != NULL) {

temp = answersHead;

answersHead = answersHead->next;

free(temp);

}

return 0;

}

// Function to play each quiz level

void playLevel(Question questions[], const char \*levelName, AnswerNode \*\*head) {

int selected[QUESTIONS\_TO\_ASK] = {-1};

int i, randIndex, score = 0;

char answer;

// Select unique random questions

for (i = 0; i < QUESTIONS\_TO\_ASK; i++) {

do {

randIndex = rand() % MAX\_QUESTIONS;

int j, found = 0;

for (j = 0; j < i; j++) {

if (selected[j] == randIndex) {

found = 1;

break;

}

}

if (!found) {

selected[i] = randIndex;

break;

}

} while (1);

}

// Ask questions

for (i = 0; i < QUESTIONS\_TO\_ASK; i++) {

printf("\nQ%d: %s\n", i+1, questions[selected[i]].question);

printf("%s\t%s\t%s\t%s\n", questions[selected[i]].options[0],

questions[selected[i]].options[1],

questions[selected[i]].options[2],

questions[selected[i]].options[3]);

printf("Your Answer (a/b/c/d): ");

scanf(" %c", &answer);

insertAnswer(head, selected[i], answer);

if (answer == questions[selected[i]].correct) {

printf("Correct!\n");

score++;

} else {

printf("Wrong! Correct answer is %c\n", questions[selected[i]].correct);

}

}

printf("\nYour Score in %s Level: %d/5\n", levelName, score);

if (score < 3) {

printf("You did not qualify for the next level. \n ---------Game Over--------!\n");

exit(0);

} else {

printf("Qualified for next level!\n");

}

}

// Function to insert answer into linked list

void insertAnswer(AnswerNode \*\*head, int qNumber, char ans) {

AnswerNode \*newNode = (AnswerNode \*)malloc(sizeof(AnswerNode));

newNode->questionNumber = qNumber;

newNode->playerAnswer = ans;

newNode->next = NULL;

if (\*head == NULL) {

\*head = newNode;

} else {

AnswerNode \*temp = \*head;

while (temp->next != NULL)

temp = temp->next;

temp->next = newNode;

}

}

// Function to display user answers

void displayAnswers(AnswerNode \*head) {

while (head != NULL) {

printf("Question %d: Your Answer = %c\n", head->questionNumber + 1, head->playerAnswer);

head = head->next;

}

}