#include<stdio.h>

#include<string.h>

//declare struct

typedef struct

{

int raceType;

char raceVenue[30];

int raceDay;

int raceMonth;

char winnerFirstName[30];

char winnerLastName[30];

float winningTime;

}Race\_t;

//declare functions

void addRace(Race\_t\* raceArr, int size);

void findBestTime(Race\_t\* raceArr, int size);

void determineEmpty(Race\_t\* raceArr, int size);

void main()

{

//variables

int i;

int maxRaces;

int menuSelect;

Race\_t\* raceDetails;

//files

FILE\* winners;

winners = fopen("WinningRace.txt", "a");

//prompt user to enter size of libary

printf("Enter Max Number of Races: ");

scanf("%d", &maxRaces);

//dynamically make 1d array of structs

raceDetails = (Race\_t\*)malloc(maxRaces \* sizeof(Race\_t));

//initalise the struct array of winning race times to -1

for (i = 0; i < maxRaces; i++)

{

//set default values to 0 or blank

(raceDetails + i)->winningTime = -1;

}

//menu to prompt user

do

{

printf("\n===============================");

printf("\nAdd New Winning Race : Enter 1 ");

printf("\nDisplay Best Time : Enter 2 ");

printf("\nOutput Empty Spaces : Enter 3 ");

printf("\nExit : Enter -1");

printf("\n==============================");

printf("\nSelect: ");

scanf("%d", &menuSelect);

//call functions

switch (menuSelect)

{

case 1:

//call add function

addRace(raceDetails, maxRaces);

break;

case 2:

//call best time function

findBestTime(raceDetails, maxRaces);

break;

case 3:

//determine empty

determineEmpty(raceDetails, maxRaces);

break;

}//switch

} while (menuSelect != -1);

//on Exit output data to file

for (i = 0; i < maxRaces; i++)

{

//output

fprintf(winners, "%d00m %s %d %d %s %s %.2f \n", (raceDetails + i)->raceType, (raceDetails + i)->raceVenue, (raceDetails + i)->raceDay, (raceDetails + i)->raceMonth, (raceDetails + i)->winnerFirstName, (raceDetails + i)->winnerLastName, (raceDetails + i)->winningTime);

}

//close files

fclose(winners);

//free Memory

free(raceDetails);

}//main

//==== myFunctions =======

void addRace(Race\_t\* raceArr, int size)

{

//local variables

int i;

int isAdded = 0; //false

//read through the race array

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

{

//check current race time

if ((raceArr + i)->winningTime == -1)

{

isAdded = 1; //true

//enter race details

printf("\nPlease Enter Race Type, 1 for 100m, 2 for 200m, 3 for 400m: ");

scanf("%d", &(raceArr + i)->raceType);

printf("\nPlease Enter Race Venue: ");

scanf("%s", (raceArr + i)->raceVenue);

printf("\nPlease Enter Race Day: ");

scanf("%d", &(raceArr + i)->raceDay);

printf("\nPlease Enter Race Month: ");

scanf("%d", &(raceArr + i)->raceMonth);

printf("\nPlease Enter Race Winner Firstname and Surname: ");

scanf("%s %s", (raceArr + i)->winnerFirstName, (raceArr + i)->winnerLastName);

do

{

//Error handling to ensure time is greater than zero

printf("\nPlease Enter Race Time: ");

scanf("%f", &(raceArr + i)->winningTime);

} while ((raceArr + i)->winningTime <= 0);

//exit loop

return;

}//if

}//for

if (isAdded != 1)

{

printf("\nCant Add Race - Memory is Full!\n");

}

}//addRace

void findBestTime(Race\_t\* raceArr, int size)

{

//local variables

char first[30];

char last[30];

int type;

int i, j;

int found = 0; //false

float currentTime;

float fastTime = 0;

int counterFast = 0;

//prompt user to enter names and type to search

printf("\nPlease Enter race winner First and Last Names: ");

scanf("%s %s", first, last);

printf("\nPlease Enter Race Type, 1 for 100m, 2 for 200m, 3 for 400m: ");

scanf("%d", &type);

//search through database for matching details

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

{

//check first name

if (strcmp((raceArr + i)->winnerFirstName, first) == 0)

{

//check last name

if (strcmp((raceArr + i)->winnerLastName, last) == 0)

{

//check race type

if ((raceArr + i)->raceType == type)

{

found = 1; //true

//search database for the fastest time

currentTime = (raceArr + i)->winningTime;

if ((fastTime < currentTime) && (counterFast == 0))

{

fastTime = currentTime; //only runs first time

counterFast++;

}

else if (fastTime < currentTime)

{

fastTime = currentTime;

counterFast++;

}

}//if

}

}//if

}//for

if (found == 1)

{

printf("\nFastest Time is: %f", fastTime);

}

else

{

printf("\nDetails Not Found!\n");

}

}//findBestTime

void determineEmpty(Race\_t\* raceArr, int size)

{

//local variables

int i;

int counter = 0;

int spaceLeft;

//read through the race array

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

{

//check if winning time has default value

if ((raceArr + i)->winningTime != -1)

{

counter++;

}

}

//calculate space

spaceLeft = size - counter;

//output

printf("\nSpace Left is: %d", spaceLeft);

}//determineEmpty