Assignment #8

Introduction to C Programming – COP 3223

Objective

1. To practice developing, writing and calling functions.
2. To practice planning and implementing complex programs.
3. To learn how to use structures as a data type

Introduction: Programmers for a Better Tomorrow

Programmers for a Better Tomorrow is an organization dedicated to helping charities, medical societies, and scholarship organizations manage various tasks so that they can focus on making the world a better place! They have asked you and your classmates to help them develop some new programs to benefit their organizations.

Problem: Run For Charity (charityrun.c)

Charity Runs are a popular way for organizations to raise awareness and get fit at the same time. However, these events can be tough to organize. The Princess Half Marathon Weekend, for example, has as many as 40,000 runners raising money and awareness for the Children's Miracle Network Hospitals. People can run both 5ks and 10ks and there's even a children's race.

After nearly a semester of C programming, you've decided that you'd like to donate your skills to create a program that organizes the typical activities for charity run weekend. In particular, your program will help manage the following:

1) Individual Registration
2) Team Registration
3) Running Events
4) Donation Totals

Your program will log the number of teams and individual who are signed up for the different races, process the racing events to see who has the fastest times, and track the total amount of money raised by teams and individuals for charity.

Header Specification

To facilitate grading, include in your header comment which portions of the assignment you have completed. You must complete all portions in order to earn full credit, but partial credit is available for completing some of the steps. The primary steps are as follows:

(1) Processing Individual Registrations
(2) Processing Team Registrations
(3) Processing the running events
(4) Calculating total donations

If your comment is accurate, meaning that you pass the appropriate tests cases corresponding to your choice, you'll earn 10 points. If your comment is roughly accurate, meaning that you sincerely

attempted the items you listed, and most of them work minus a tiny bug, then you'll get 5 of these points. If your comment isn't accurate to a reasonable degree, you'll get 0 of these points.

The reason this is here is because it's very important to communicate to others accurately what you've accomplished and what is left to accomplish. This sort of honesty and ability to appraise your own work is a critical skill in a job.

Program Details

Individual Registration Details

There are a two registration windows: early registration and regular registration. Prices for registering early and regularly will be given. Each individual will have a unique number and must be processed separately to record their name, age, running event, and donations raised. The maximum number of runners for our program will be 10000.

Team Registration Details

Teams may be created and registered to sign up several participators at once. Teams may only sign up during early registration, have between 5 and 50 members, and must pay the team registration fee for every member on the team. Teams should be recorded with their name and number of members. Each member of the team still needs to be recorded with their name, age, running event, and donations raised. We can organize at most 200 teams.

Running Events Details

There will be three running events: a 5k race, a 10k race, and a marathon race. For each race, your program will receive the running time for each participant in the race. For the 5k and the 10k you should print the runner with the fastest time. For the marathon, your program will need to check times against the required qualifying times to see which runners qualify for more marathon races. These qualifying times vary by age group.

Total Donation Details

After all the races are run we want to recognize the participants who raised the most money for charity. Print the team that raised the most money for the event along with the amount raised. Then for each team, print the team member who raised the most money along with the amount raised. For the individuals, print the top individual who raised the most money along with the amount raised. Finally, print the total amount raised for the event. All donations and registration fees will be donated directly to charity.

Implementation Restrictions

You must use the following constants:

```
#define TEAMS 200
#define RUNNERS 10000
#define LENGTH 20
#define TEAMSIZE 50
```

You must use the following structures to store information:

```
struct person {
        char name[LENGTH];
        int number;
        int age;
```

```
        int event;
        float money;
        float time;
};

struct team {
        char name[LENGTH];
        int nummembers;
        float money;
        struct person members[TEAMSIZE];
};
```

It is not sufficient to simply have the structures in your program, you must use them store information and process operations for the program.

Though function prototypes will not be provided, it is expected that you follow good programming design and create several functions with well-specified tasks related to the solution of this problem. Make sure to pay very careful attention to parameter passing.

Input Specification

The first line of the file contains the following three values, separated by spaces:

> Cost of early registration for individuals (in dollars), Cost of regular registration for individuals (in dollars), and the cost of team registration (in dollars). These will be positive real numbers to two decimal places.

The second line of the file will contain one positive integer representing the number of early individuals and teams who are registering. Every registration will start with either TEAM or INDV for a team registration or individual registration respectively.

Lines that begin with INDV will be followed by four values, separated by spaces:

The individual's name, their age, their chosen running event, and the amount of donations they have raised. The first is a string, the second is a positive integer, the third is an integer from the set {5, 10, 42}, and the fourth is a positive real number.

Lines that begin with TEAM will be followed by two values, separated by spaces: the name of the team and the number of team members (k). This will be followed by k lines organized the same as the individual registration above.

After early registration completes, normal registration can begin. This will be represented by one positive integer (m) representing the number of regular registrations. All teams must use early registration. This will be followed by m lines each with four values. These are the same values that are present in individual registration: The team member's name, their age, their chosen running event, and the amount of donations they have raised. The first is a string, the second is a positive integer, the third is an integer from the set {5, 10, 42}, and the fourth is a positive real number.

After registration, the running events can occur. Every registered participant will be listed with their number (assigned as part of registration) and the time it took them to run the race in minutes represented as a real number.

This will be followed by 10 lines of qualifying times based on age groups. They will be specified:

STARTINGAGE ENDINGAGE TIME

where starting age and ending age are integers, and the qualifying time is a real number representing minutes.

Output Specification

For an individual registering for an event print a line of the form:

X registered for the Y race! They have been assigned the number Z.

where X is their name and Y is the event they are registering for. Y is represented by an integer {5, 10, 42} in the input file. Replace it with "5k" for the first integer, "10k" for the second integer, and "marathon" for the third integer in this print statement. Z is their unique runner number. These numbers should start at 1 and count up to a max of 10000.

For a team registering print a line with one of the form:

X team registered for the event. They have Y members:

where X is the team name and Y is their number of members. Follow this with Y lines of the same form as the individuals: one for each member of the team.

For the 5k race and the 10k race, output a single line of the following form:

Xk race: Y had the fastest time with Z.Z minutes!

where X is either 5 or 10 respectively, Y represents the name of the fastest runner and Z represents their time.

For the marathon race, print out all the runners who times meet the qualifying times:

X qualified in the marathon run with a time of Y.Y minutes!

where X represents the name of the fastest runner and Y represents their time.

For the team that raised the most money, output a single line of the following form:

The X team raised the most money with a team donation of $Y.YY!

where X is the team name and Y is the amount they raised.

For each team, print out the person that raised the most with a single line of the following form:

X raised the most money on team Y with a donation of $Z.ZZ!

where X is the person's name, Y is the team name, and Z is the amount they raised.

For the individual that raised the most money, output a single line of the following form:

X raised $Y.YY!

where X is the person's name and Y is the amount they raised.

End with the total amount raised by the event:

The runners raised $X.XX for charity!

Sample Input/Output

Four sets of input and output are provided with the assignment, showing the different portions of the assignment. You should try to complete race01 first, then proceed to race02, and so forth.

Each test file is labeled as raceXX.txt for input and raceXX.out for output. Out files can be opened in notepad and other unformatted text editors.

Deliverables

A single source file named *charityrun.c* turned in through WebCourses.

Restrictions

Although you may use other compilers, your program must compile in gcc and run in Code::Blocks. Your program should include a header comment with the following information: your name, course number, section number, assignment title, and date. Make sure you include comments throughout your code describing the major steps in solving the problem.

Make sure to use good programming style, including use of appropriate constants, good variable names and good use of white space.

Grading Details

Your programs will be graded upon the following criteria:

1) Your correctness

2) Your programming style and use of white space. Even if you have a plan and your program works perfectly, if your programming style is poor or your use of white space is poor, you could get 10% or 15% deducted from your grade.

3) Compatibility – You must submit C source files that can be compiled and executed in a standard C Development Environment. If your program does not compile, you will get a sizable deduction from your grade.

4) Header Comment - In your header comment, you must state which portions of the assignment you have implemented. Your choices are as follows:

(1) Processing Individual Registrations
(2) Processing Team Registrations
(3) Processing run events
(4) Calculating total donations

If your comment is accurate, meaning that you pass the appropriate tests cases corresponding to your choice, you'll earn 10 points. If your comment is roughly accurate, meaning that you sincerely attempted the items you listed, and most of them work minus a tiny bug, then you'll get 5 of these points. If your comment isn't accurate to a reasonable degree, you'll get 0 of these points.

The reason this is here is because it's very important to communicate to others accurately what you've accomplished and what is left to accomplish. This sort of honesty and ability to appraise your own work is a critical skill in a job.