Introduction to C - Programming Assignment #2

Due date: September 23, 2011, 11:59pm

Objectives

1. To learn how to use an if statement for conditional execution.
2. To learn how to call math library functions to aid in calculations.
3. To learn how to use basic repetition to simplify calculations and allow for statements to repeat certain sets of statements multiple times.

Problem A: Wedding Invitations (invite.c)

Though Arup wants to just send a free E-vitation to all of his guests for his wedding, his fiancee will have none of it. She has put her foot down and has insisted that they mail out regular invitations. After doing some research, Arup found out that depending on the size of the packages of invitations ordered, there are different prices. He wants you to figure out how many of each package to order, in order to get enough invitations to mail out at the cheapest price. For the purposes of this problem, there are only two packages:

1) One with 50 invitations
2) One with 200 invitations

Each vendor has a different price for the two packages. You'll ask the user to input these two prices, as well as the total number of invitations that need to be sent out. Your job will be to calculate how many of each package to buy for the best deal, and how much will be spent on invitations.

Input Specification

1. The cost of both packages (in dollars) will be positive real numbers less than 500.
2. The number of invitations needed will be a positive integer less than 10000.

Output Specification

The first line of output will specify the number of small packages to buy to minimize the cost of the order of invitations using the following format:

```
You should order X small package(s).
```

where X is the number of small packages to order.

The second line of output will specify the number of large packages to buy to minimize the cost of the order of invitations using the following format:

```
You should order Y large package(s).
```

where Y is the number of large packages to order.

The last line of output should include the total cost of the packages using the following format:

```
Your cost for invitations will be $Z.
```

where Z is the minimum cost in question outputted to two decimal places.

Output Sample

Below is one sample output of running the program. **Note that this sample is NOT a comprehensive test.** You should test your program with different data than is shown here based on the specifications given above. In the sample run below, for clarity and ease of reading, the user input is given in *italics* while the program output is in **bold**. (Note: When you actually run your program no bold or italics should appear at all. These are simply used in this description for clarity's sake.)

Sample Run #1

```
What is the cost of a small package (in dollars)?
50.99
What is the cost of a large package (in dollars)?
150.99
How many invitations are you sending out?
240

You should order 1 small package(s).
You should order 1 large package(s).
Your cost for invitations will be $201.98.
```

Note: There are actually many somewhat tricky test cases for this problem. Keep in mind that the cost of the two packages might be anything. In some cases you might go with all of one package. In other cases, you might need to go with only the other package. Finally, in cases like this sample, a mixture of the packages is necessary.

Problem B: New Home (home.c)

One of the issues that Arup and his fiancee have had in their wedding plans is figuring out where to live after they get married. They each have different places they visit frequently (their parents, work, etc.) and want to minimize the amount of driving they do to be fair to each other. Given a set of three points that they visit frequently, it turns out that the centroid between the three points determines a good central location to live. (Go online to find the definition of the centroid of a triangle and how to find its coordinates.)

In this problem, you will be given the x and y coordinates of three locations Arup and his fiancée want to be close to. Your job will be to calculate the following:

a) The coordinates of the centroid of the three given points.
b) The distances from the centroid to each of the three original points in question.

Input Specification

All three sets of x-y coordinates will be real numbers in the range [-50,50], where the numbers represent miles.

Output Specification
The first line of output will have the following format:

```
You should live at location (X, Y).
```

where X and Y are printed to two decimal places.

The second line of output should have the following format:

```
The corresponding travel distances are A, B and C.
```

where A, B and C represent the distances from the centroid to the three original points, listed in increasing order by distance, printed to two decimal places (in miles).

Output Samples
A sample of the program running is included below. **Note that this sample is NOT a comprehensive test.** You should test your program with different data than is shown here based on the specifications given above.

Sample Run

```
What are the x-y coordinates of the first location?
2.0 5.6
What are the x-y coordinates of the second location?
3.1 -4.7
What are the x-y coordinates of the third location?
-2.2 3.5
You should live at location (0.97, 1.47).
The corresponding travel distances are 3.76, 4.26 and 6.53.
```

Problem C: Infinite Cake (cake.c)
Wedding Cakes come in many designs. One of the designs is a set of cylindrical layers. Each layer, from top to bottom has a bigger radius than the one above it. The number of people each layer feeds is based up on the area of the full top surface of the cake. In order for the cakes to look good, the sequence of radii of the different layers must follow an arithmetic sequence with a positive difference. (If you forgot what an arithmetic sequence is, please go online and look this up.)

Your problem will be to figure out how many people a particular cake design can feed, based on the number of layers the cake has, the radius of the top layer cake, the radius of the bottom layer cake, and the amount of cake (described by the area of the top of the piece) that one person eats.

For example, if a layer of cake has cross-sectional area 30.7 units^2 and each person eats 5.3 units^2 worth of cake, then 30.7/5.3 = 5.79 people can be fed. BUT, people aren't fractional and we can't give someone parts of cake from two different layers. Thus, in

this case, we'll say that the cake feeds 5 people and each of those five pieces is a bit bigger than necessary.

To correctly solve the problem, make this calculation separately for each layer. For example, if one layer feeds 5.79 people and another layer feeds 8.66 people, together they would only feed 5 + 8 = 13 people, instead of 5.79 + 8.66 = 14.45 or 14 people.

Input Specification

1. The number of layers will be a positive integer less than 100.
2. The radius of the top layer will be a positive real number less than 1000.
3. The bottom layer radius will be a positive real number at least as big as the top layer and less than 1000. (In a one layer cake, this number will be the same as the top layer radius.)
4. The amount of cake necessary for each individual will be a positive real number representing cross-sectional area less than the cross-sectional area of the top layer.

Output Specification

Output the total number of guests that can be fed with the given cake design:

```
Your cake will feed X guests.
```

where X is the number of guests in question.

Output Samples

A sample output of running the program is included below. **Note that this sample is NOT a comprehensive test.** You should test your program with different data than is shown here based on the specifications given above.

Sample Run #1

```
How many layers will your cake have?
5
What is the radius of the top layer?
6
What is the radius of the bottom layer?
14
How much cake in cross-sectional area does each guest need?
4.5
Your cake will feed 374 guests.
```

Deliverables

Three (or four, if you do the extra credit) source files:

1) *invite.c*, for your solution to problem A
2) *home.c* for your solution to problem B
3) *cake.c* for your solution to problem C

All files are to be submitted over WebCourses.

Restrictions

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

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 to gcc in Code::Blocks. If your program does not compile in this environment, you will get a **sizable** deduction from your grade.