CSCI 1312 (Introduction to Programming for Engineering), Fall 2017

Homework 3

Credit: 35 points.

1 Reading

Be sure you have read (or at least skimmed) the assigned readings from chapter 5.

2 Honor Code Statement

Please include with each part of the assignment the Honor Code pledge or just the word "pledged", plus one or more of the following about collaboration and help (as many as apply).[1] Text *in italics* is explanatory or something for you to fill in. For written assignments, it should go right after your name and the assignment number; for programming assignments, it should go in comments at the start of your program(s).

- This assignment is entirely my own work. *(Here, "entirely my own work" means that it's your own work except for anything you got from the assignment itself — some programming assignments include "starter code", for example — or from the course Web site. In particular, for programming assignments you can copy freely from anything on the "sample programs page".)*
- I worked with *names of other students* on this assignment.
- I got help with this assignment from *source of help — ACM tutoring, another student in the course, the instructor, etc. (Here, "help" means significant help, beyond a little assistance with tools or compiler errors.)*
- I got help from *outside source — a book other than the textbook (give title and author), a Web site (give its URL), etc.. (Here too, you only need to mention significant help — you don't need to tell me that you looked up an error message on the Web, but if you found an algorithm or a code sketch, tell me about that.)*
- I provided help to *names of students* on this assignment. *(And here too, you only need to tell me about significant help.)*

3 Programming Problems

Do the following programming problems. You will end up with at least one code file per problem. Submit your program source (and any other needed files) by sending mail to `bmassing@cs.trinity.edu` with each file as an attachment. Please use a subject line that mentions the course and the assignment (e.g., "csci 1312 hw 3" or "CS1 hw 3"). You can develop your programs on

[1] Credit where credit is due: I based the wording of this list on a posting to a SIGCSE mailing list. SIGCSE is the ACM's Special Interest Group on CS Education.

any system that provides the needed functionality, but I will test them on one of the department's Linux machines, so you should probably make sure they work in that environment before turning them in.

Some words about checking for non-numeric input: Doing a great job of "validating input" is probably outside the scope of this course — it turns out to be not an easy problem to solve in C — so in the programs you write for homework, generally speaking what I have in mind is for you to just check the return value from `scanf`, as I do in the examples in class, and bail out of the program if it indicates an error. You may be tempted to make your program prompt the user again, but don't do that either — it breaks my semi-automated testing and again is not trivial to do in C. Just checking the return value from `scanf` is far from perfect but at least ensures that the variable(s) read from standard input have sensible values, which is good enough for our purposes right now. Note that if the problem doesn't make sense for some inputs (e.g., "counting change" for a negative number) I'll usually also ask that you reject such input (with an error message).

1. (15 points) Write a C program that asks the user for three integers and prints them in order from smallest to largest, or an error message if what was entered is something other than three integers. (This will turn out to be a special case of a more general problem of putting things in order. Don't try right now to solve the general problem; just solve it for three numbers using what we've discussed in class up through 9/13.) Notice that the numbers do not have to be distinct — the input could be three values all the same, or two the same and one different, or all three different.

2. (20 points) For this problem, imagine a company (call it Acme Caps) that makes and sells custom baseball caps, priced as follows:

   - The minimum order is 50 caps, for which the price is $100. Acme will fill orders for fewer caps, but will still charge $100 for them.
   - For more than 50 caps, Acme charges the minimum of $100 for the first 50 caps, plus $5 for each additional group of 5 caps (rounding up if the number is not a multiple of 5). (So for example 55 caps cost $105, and for 51 caps the price is the same.)
   - Orders of more than 200 caps get a volume discount; Acme charges its minimum ($100) for the first 50 caps, $5 per group of 5 for the next 150 caps, and then only $4 for each group of 5 caps beyond the 200-cap threshold. (So for example 205 caps cost $254, and for 201 caps the price is the same.)

   (Is this a realistic business model? I don't know! but I think it makes a reasonable programming problem for this assignment.)

   Write a C program that prompts the user for the number of caps and prints the corresponding price. Of course(?), the program should print an error message and stop if input is not numeric or numeric but less than 0. The program should also print an informational message if the price would be the same for a larger number of caps (i.e., the number entered is less than the "minimum" order, or not a multiple of 5). It should also print a message if the order qualifies for the volume discount.

   Some sample executions (`[prompt]` is the usual prompt, usually your user name plus machine name, etc.)

```
[prompt]$ ./a.out
how many caps?
45
price $100
you could get 50 caps for the same price

[prompt]$ ./a.out
how many caps?
80
price $130

[prompt]$ ./a.out
how many caps?
82
price $135
you could get 85 caps for the same price

[prompt]$ ./a.out
how many caps?
101
price $155
you could get 105 caps for the same price

[prompt]$ ./a.out
how many caps?
200
price $250

[prompt]$ ./a.out
how many caps?
202
you qualify for the volume discount!
price $254
you could get 205 caps for the same price

[prompt]$ ./a.out
how many caps?
don't know
invalid input (not integer)

[prompt]$ /a.out
how many caps?
-12
invalid input (not positive)

[prompt]$ ./a.out
how many caps?
0
```

```
invalid input (not positive)
```