CIS 130 - Intro to Programming - Fall 2005
Homework Assignment #7
(#5 REPHRASED 10-24-05)

Homework #7:
HW #7 PART 1 is due by **12:00 NOON** on **Wednesday, October 26, 2005;**
HW #7 PART 2 is due by **12:00 NOON** on **Friday, October 28, 2005**

Week "9" Lab Exercise - due at end of your registered lab section
on either 10-21 or 10-24

WEEK "9" LAB EXERCISE

1. Consider the following function:

```
// Contract: compare2: double double -> int
// Purpose: return -1 when <val1> is less than <val2>,
//          return 0 when <val1> equals <val2>, or
//          return 1 when <val1> is greater than <val2>.

// Examples: compare2(3, 4) _____________________________________

//           compare2(7, 7) _____________________________________

//           compare2(6, 2) _____________________________________

int compare2(double val1, double val2)
{
    if (val1 < val2)
    {
        return -1;
    }
    else if (val1 = val2)
    {
        return 0;
    }
    else
    {
        return 1;
    }
}
```

The intent of this function is to return **-1** when its first argument is strictly less than its second, to return **0** when the two arguments are equal, and to return **1** when the first argument is strictly greater than the first.

(a) Given the above description of what this function SHOULD be doing, you should be able to complete its "Examples" section. Do so, using the "new" preferred notation.

However --- a semantic/logic error lurks in the actual code above. When you run the given

examples, you see that:

```
compare2(3, 4)        ...actually returns -1
compare2(7, 7)        ...actually returns 0
compare2(6, 2)        ...actually returns 0
```

(b) Which of these tests is NOT returning the "correct" value?

_____

(c) What, then, is wrong with this function, as currently written? Determine the problem, fix it, and then type in this function using **funct_play1** or **funct_play2** to see if your solution fixes the problem.

When it is time to get your lab work checked, I'll check your written answers to parts **(a)** and **(b)**, look at your modified function on cs-server, and have you run the three examples to show me how you did at correcting the problem.

2. Consider the following:

```
double x(double a,double j,double q){if
(((a<=j)
&&(a<=q))){return a;}else
if ((a>=q) && (j>=q)) {return
q; } else if ((a>=j) &&
(j<=q)) {return j;}}
```

(a) This is a working function; its logic is actually quite sound. Its style is very poor, however. What does it do?

_____

_____

_____

_____

(b) Rewrite this so that it meets course coding standards. Then type it in using **funct_play2** (including all of the proper design recipe steps), and test it on all of your examples.

When it is time to get your lab work checked, I'll check your written answer to part **(a)** and look at your re-written version of the the function on cs-server.

When you have completed all of the above, write your name on the 'Next:' list on the board to get your work checked. All of the above must be completed before the end of your lab time.

HOMEWORK #7

You are to work **individually** on this assignment.

PART 1: (30% of the HW #7 grade) Create a file **130hw7_part1.txt** on cs-server. Inside this file, type the following:

* your **Contracts** for the functions you are to write for the problems below.
* your **Examples** parts of the design recipe for the problems below.

(NOTE: I am expecting that you'll be writing the headers and purpose statements in between writing the contracts and the examples that you submit here; I am just not requiring you type them into **130hw7_part1.txt**.)

You must submit this file using ~st10/130submit (typed at the cs-server prompt, from the directory where your 130hw7_part1.txt file resides!!) by **12:00 noon** on **Wednesday, October 26th** to receive **any** credit for Part 1 of HW #7.

PART 2: (70% of the HW #7 grade) Finish the problems below. You must enter your functions using an appropriate choice of the **funct_play*** tools (remembering that all design recipe elements ARE expected and required), and test them using either **funct_play*** or **expr_play**.

When you are ready, you must submit:
* all of your **.cpp** and **.h** files for the problems below;

...using **~st10/130submit**. These must be submitted by **12:00 noon** on **Friday, October 28th** to receive **any** credit for Part 2 of HW #7.

1. Write a simple warm-up function **my_abs**. It takes a numeric argument, and returns the absolute value of that argument. You are required to solve this using an appropriate **if** statement (you may not call cmath's **abs** function); the purpose of this is to give you a simple function using **if** to "warm up" with.

2. Write a function **my_max**. It takes two numeric arguments, and returns the value of the larger of the two arguments. (If the values are equal, then the function should simply return that common value.) Again, you are required to solve this using an appropriate **if** statement (you may not call cmath's **max** function); this is another simple warm-up.

3. A classic if-exercise is to write a function **max3** --- it takes 3 values, and returns the largest value of the three (and the common value if all are the same, or the common highest value if two are equal and higher than the third).

We're going to do it a little differently, however --- you are required to make some appropriate use of **my_max** in your solution. If you do this reasonably, you will find that it significantly simplifies your solution (as compared to a version that does not use **my_max**).

4. (Note: this is NOT a conditional function. It is an auxiliary function for the next problem.) A local discount store has a policy of putting labels with dates on all of its new merchandise.

Write a small function **num_weeks** that accepts as its argument the number of days since an item has arrived; it returns the number of weeks that the item has been in the store, but it always returns it as a whole number: for example, if it has been 6 days, it should return 0, saying that the item hasn't been in the store for a whole week yet. If it has been 14 days, it would return 2 --- it has now been in the store for two weeks. And, for 22 days, it would return 3 --- it has been in the store for over 3 weeks, but not yet for 4 weeks.

IMPORTANT NOTE: given the types that you *should* be using for this function, note that the cmath **floor** function should not be a good choice. (If you use the types you should use, you will get error messages if you try to use floor.) However, if you use the types you should use, the division you will be using should also be just right --- and that's a very big hint!!

5. (Adapted from www.htdp.org) We're still considering the same local discount store as in problem #4. Now, we are considering its pricing policy.

rephrased description:
Until the item has been in the store for two weeks, it is sold for original price. Once it has been in the store for two weeks --- but not yet three --- it is sold for 25% off (its original price is reduced by 25%). If an item has been in the store for three weeks --- but not yet four --- it is sold for 50% off (its original price is reduced by 50%). And, if it has been in the store for four weeks or more, it is sold for 75% off (its original price is reduced by 75%).

original description:
For the first two weeks, an item sells at its original price. If it still has not been sold after the first two weeks, the store discounts the item's original price by 25% during the third week. If it still has not been sold after the first three weeks, the store discounts the item's original price by 50% during the fourth week. And, if it still has not been sold after the first four weeks, the store discounts the item's original price by 75% during the fifth week and beyond (after that no additional discounts are given).

(If you'd like to reconcile these two descriptions, note that the idea is that the first two weeks is before the item has been in the store for two full weeks. The third week is when the item has been in the store for two full weeks, but not yet three; the fourth week is when the item has been in the store for three full weeks, but not yet four. But if you don't care about reconciling the two descriptions, just go with the rephrased one; I think it is easier to follow. Maybe.)

Write a function **new_price**, which expects as its arguments the initial/original price of an item and the number of days since the item was dated; it should return the current selling price of the item. You are required to make reasonable use of your function **num_weeks** from problem #4.

6. (Adapted from www.htdp.org) A manufacturing company measured the productivity of its workers and found that between the hours of 6 am and 10 am --- that is, from 6-7, 7-8, 8-9, and 9-10 --- they could produce 30 pieces/hour/worker;

between 10 am and 2 pm --- from 10-11, 11-12, 12-1, and 1-2 --- they could produce 40 pieces/hour/worker;

and between 2 pm and 6 pm --- from 2-3, 3-4, 4-5, and 5-6 --- they could produce 35 pieces/hour/worker.

At all other hours, 0 pieces are produced per hour per worker --- the company is closed then.

Write a function **pieces_produced** that takes as its arguments an hour of the day expressed in twenty-four hour format, along with the number of workers working during that hour; it computes and returns the total number of pieces produced by that many workers during that hour of the day. (A first argument of 15, then, is asking how much the given number of workers produces working from 15:00-16:00, or from 3-4 pm.)