CS367 Lab 6

This lab consists of three parts. For Part **A** you will complete the attached worksheet and submit it as *hardcopy* by the start of class on the due date. For Parts **B** and **C** you will write a complete C program following the instructions for program organization and **Moodle** submission given in Lab **5**, with one change: in addition to the submission files specified in Lab **5** each program directory should contain a **Makefile** that you used to build your program. Your **Makefile**'s can be based on either the basic or the advanced **Makefile**'s shown in class.

Part A. Refer to the attached worksheet. Assume that the program is running and that execution has reached the point marked **HERE** in the function named **second**. On the worksheet draw the current memory state for all variables and arguments in all three functions (except for the **main** arguments **argc** and **argv**). Use the simplified "boxes and arrows" approach that Kevin used in lecture. Variables and arguments that are local to each function should appear in the rectangle associated with that function; arrows may/should extend between variables in different functions as appropriate. *No variables or arguments should appear outside of a function memory rectangle.*

Part B. For this Part you will write a program consisting of the following two functions:

1. A function that takes as arguments two integers, a *value* and a *maximum allowable value*. The function should ensure that the value is no greater than the maximum allowable value by doing the following: If the value is greater than the maximum allowable value, the value should be set to the maximum allowable value. The function should return true if the value was changed, and false if the value was not changed (i.e., if the value was less than or equal to the maximum allowable value). *Hint:* for each argument ask: can it be changed by the function?

2. A **main** function that does the following:

a. Get two integers, a value and a maximum allowable value, from the user. The user should be given as many chances as necessary to input valid integers for both values (*hint:* this is easier if you have the user enter each integer using separate input statements).

b. Call your step **1** function with the user-input value and maximum allowable value.

c. Output whether or not the value was changed by being passed as an argument in step **b**.

d. Output the following, clearly labeling each number: the maximum allowable value, the original user input value, and the final value (after step **b**).

Test your program with the following inputs and submit the output along with your source code files and your **Makefile**.

Run 1:

value:	**50**
max value:	**25**

Run 2:

value:	**abc**
corrected to:	**12**
max value:	**h**
corrected to:	**12**

Part C. For this Part you will write a program that calculates and outputs information concerning car fuel costs. Your program should consist of the following functions:

1. A function that takes as arguments a floating-point price-per-gallon and an integer tank capacity (in gallons) and returns the floating-point cost of filling that tank, assuming it is empty.

2. A function that takes as arguments a floating-point price-per-gallon, an integer tank capacity, and a floating-point tank fill-up cost. The function should output the three values, neatly formatted and clearly labeled. All floating-point values should be formatted as money, with a preceding '**$**' and two digits to the right of the decimal point.

3. A function that takes as arguments a floating-point price-per-gallon, an integer tank capacity, and a floating-point tank fill-up cost. The function should set the value of the tank fill-up cost argument using a call to function **(1)** with the passed-in price-per-gallon and tank capacity values.

4. A function that takes two arguments, a floating-point price-per-gallon and an integer tank capacity. The function should set each of these variables to positive values input by the user, giving the user as many attempts as needed to input valid, positive values.

5. A **main** function that does the following in the indicated order. Your **main** function should not directly perform input or output (i.e., **main** should not contain **printf**'s or **scanf**'s).

> **a.** Get a price-per-gallon and tank capacity from the user by calling your function **(4)**.
>
> **b.** Calculate the tank fill-up cost by calling your function **(3)** (*not* function (**1**)).
>
> **c.** Output all values by calling your function (**2**).

Test your program with the following inputs and submit the output along with your source code files and **Makefile** (as per Lab **5**).

price-per-gallon:	**abc**
corrected to:	**-3.5**
corrected to:	**def**
corrected to:	**4.05**
tank capacity:	**12**

Name: ________________________________

Lab 6 Part A Worksheet. ***See instructions on main assignment sheet.***

```
int main (int argc, char* argv[])
{
      float x = 5;
      float* xp = &x;
      doit(*xp);
      return 0;
}
```

main *function memory*

```
void doit(float y)
{
      float* yp = &y;
      second(yp);
}
```

doit *function memory*

```
float second(float* a)
{
      float tmp = *a;
      *a = tmp + 6;
———————— HERE ——————————
      return tmp;
}
```

second *function memory*