IPC144 Lab #6 - More Functions (Mr. Saul's Sections)

Weekly Tip: A function that is passed an address receives the address in a pointer variable. The scope of that pointer variable extends from its declaration as a parameter to the end of the function.

In this lab, you will work with functions that are passed addresses. As with the previous labs, if you get stuck, don't be shy about asking for help - that is what the lab sessions are for.

NOTE: When you have completed this lab, and placed answers into the file called "lab6.txt", follow the instructions (at the bottom of this lab) to submit your answers to Murray Saul's e-mail account...

To begin with, enter the following program (call it lab6a.c):

```
#include <stdio.h>
int triple1(int x) {
   return 3 * x;
}
void triple2(int x) {
   x = x * 3;
}
void triple3(int *p) {
   *p = *p * 3;
}
main() {
   int n;
   n = 5;
   printf("Before triple1, n is %d\n", n);
   n = triple1(n);
   printf("After triple1, n is %d\n", n);
   triple2(n);
   printf("After triple2, n is %d\n", n);
   triple3(&n);
   printf("After triple3, n is %d\n", n);
}
```

1. Compile this program and run it. Make sure that you relate the output to what is going on, and that you understand the answers to the following questions:

   What is the value of n after triple1 is called? ______________

   What is the value of n after triple2 is called? ______________

   What happens to the tripled value of x in triple2? __________________________

   What is the value of n after triple3 is called? __________________________

   Both triple1 and triple3 accomplish the same thing in this program. Which function do you prefer? _____ Why? _________________

   In general, the technique used in triple1 is preferred to the one used in triple3, all other things being equal, because it is more flexible from the calling function's point of view. For example, a program calling triple1 with code like:

   ```
   a = triple1(triple1(5 * b - 45) + 4);
   ```

   while a program calling triple3 to achieve the same results would require code like:

   ```
   a = 5 * b - 45;
   triple3(&a);
   a = a + 4;
   triple3(&a);
   ```

2. Copy lab6a.c to lab6b.c. In lab6b.c, remove the "&" in main's triple3 call, so that it reads

   ```
   triple3(n);
   ```

   then compile the program. What happens this time? ____

   Why? ____

3. Copy lab6a.c to lab6c.c. In lab6c.c, remove the pointer resolution asterisks from the line in triple3, so that that the line reads

   ```
   p = p * 3;
   ```

   What happens this time? ________________________________

   Why? ________________________________

4. Now enter the following program (call it lab6d.c):

   ```
   #include <stdio.h>
   int gettime1(void) {
      int h, m;
      printf("Enter a time (e.g. 12:45) - ");
      scanf("%d:%d", &h, &m);
      return 100*h + m;
   }
   void gettime2(int *ph, int *pm) {
      printf("Enter a time (e.g. 12:45) - ");
      scanf("%d:%d", ph, pm);
   }
   main() {
      int hr, min, time;
      time = gettime1();
      printf("You entered %d:%02d\n",time/100,time%100);
      gettime2(&hr, &min);
      printf("You entered %d:%02d\n", hr, min);
   }
   ```

   Compile and run this program. Again, make sure that you relate the output to what is going on.

   What technique does gettime1 use to return 2 pieces of information?

   ____

   Note how anything in the scanf format string other than a %-specification is **fixed** data that the user MUST enter, but is ignored by scanf when it is filling memory locations. For example, in this case, if the user enters 10:45, scanf will place 10 in the first variable, 45 in the second variable and will simply ignore the ":". But if the user fails to enter the ":" between the two numbers, scanf will fail.

   Why does the call to scanf in gettime2 work even though there are no ampersands (&) before the variable names ph and pm?

   ____

   What technique did gettime2 use to transfer data values to the main function?

   ____

   Both gettime1 and gettime2 accomplish the same thing in this program. Which do you prefer?

   ____

   In this case, the technique used in gettime2 is generally preferred, since it avoids a multiplication and a division, and since it is not always possible (depending on the nature of the data) to "combine" two values into one as gettime1 does.

5. [If you have time] Write a function:

   ```
   int gettime(int *ph, int *pm)
   ```

   that works like gettime2 but also returns the value 1 if the hour is between 1 and 12 and the minute is between 0 and 59, or returns the value 2 if the hour is between 0 and 23 (but not between 1 and 12) and the minute is between 0 and 59. If neither of these conditions is met by the numbers entered, then the function returns the value 0.

   Write a main that uses gettime to first get a time on a twelve-hour clock, making the user re-enter an invalid time, and then does the same thing for a 24-hour clock time.

<u>Submission Requirements:</u>

If you are in Murray Saul's class, issue the following command to send your lab #6 answers to Murray Saul:

```
mail -s "144lab6" -c $USER@learn.senecac.on.ca murray.saul@senecac.on.ca < lab6.txt
```

The option **-s "144lab6"** makes subject line appear as "144lab6" so instructor can filter these e-mails in a directory to collect all lab6 submissions.

The option **-c $USER@learn.senecac.on.ca** sends a copy of the e-mail message to YOUR learn account. The variable **$USER** is your Matrix id name assuming that you are issuing this command when logged into your Matrix account. Please keep this e-mail for the remainder of this term as proof that you sent your lab by the required deadline...