

**Due: 4-16-2014 (April 16,2015)**

Filename: **hello.c, inchtocm.c, interest.c, squares.c, average.c**

Add your Name and UC Davis ID number as the first comment of each file to make it easier for TA's to identify your files.

All programs should compile with no errors. Warnings are okay, as long as execution is still correct. Compile programs using "gccx filename.c", then execute with "./a.out" or specify your executable filename during compilation using "gccx -o outputFile filename.c" and execute with "./outputFile".

Use *diff* with the \*\_output files to compare the output of your executable files with the official output files. (Syntax: "diff file1 file2") *diff* will compare two files line by line, and list the changes that need to be made to the first file to make it identical to the second. If there are no differences between the two files, then *diff* outputs nothing.

### Submitting files

Usage: handin cs30 assignmentName files...

The assignmentNames are Proj1, Proj2, Proj3, Proj4, Proj5. **This is Proj1**. Do not hand in files to directories other than Proj1, they will not be graded. The files are the names of the files you wish to submit. For this assignment, your files will be hello.c, inchtocm.c, interest.c, squares.c, average.c. Do not name the file anything else. Therefore,

you would type from the proper directory:

**\$ handin cs30 Proj1 programName.c**

**Do NOT zip or compress the files in any way.** Just submit your source code.

**\$ handin cs30 Proj1 hello.c inchtocm.c interest.c squares.c average.c**

If you resubmit a file with the same name as one you had previously submitted, the new file will overwrite the old file. This is handy when you suddenly discover a bug after you have used handin. You have up to 4 late days to turn in files, with a 10% deduction for each late day. The deadline for each project (and late day) is 11:59pm.

*TA comments:*

*Some of you have been enabling c99 in your compilers so you can do things like:*

*for (int i=0; i< NUM; i++)*

*instead of:*

*int i;*

*for (i=0;i<NUM;i++)*

***Do NOT do this.*** For consistency we will not be enabling c99 during your grading scripts, so this will result in an error and you **WILL LOSE POINTS**. Please just write the extra line of code.

### **OUTPUT FILES and DIFF:**

*For this project we are giving .out files for all the problems, and input files when needed. These are EXECUTABLE FILES, NOT TEXT files. You will need to pipe the outputs of these files to an output text file as well as the outputs of your programs into a separate output text file for diff comparison.*

*For example:*

If you are working on hello.c:

Assume the official provided file is **hello\_output**, and your hello.c file generates **hello**.

To compare the outputs of the two .out files, you need to:

```
$ ./hello > myHello.txt
$ ./hello_output > hello.txt
$ diff myHello.txt hello.txt
```

**If there were no differences, nothing will print when you run ‘diff’. If you get a “Permission denied” error for any of the files (ex. hello\_output), try `$ chmod 777 hello_output` in the console/terminal.**

For the problem with required input file, do the following (ex. inchtocm.c):

```
$ ./inchtocm < inchtocm_input > myInchtocm.txt
$ ./inchtocm_output < inchtocm_input > inchtocm.txt
$ diff myInchtocm.txt inchtocm.txt
```

```
$ diff -w myInchtocm.txt inchtocm.txt (ignore differences of white spaces) - used for grading
```

**Checking with diff is very important as we will be using scripts to grade your homeworks, and even the smallest difference will result in your not getting points for a problem.**

*FOR THE GRAPHICS PROGRAMS: All the graphics programs are optional. You should still do them, but they will NOT be graded. Each program that is optional will have the word **(OPTIONAL)** next to it. **There are no graphics programs in Project 1.***

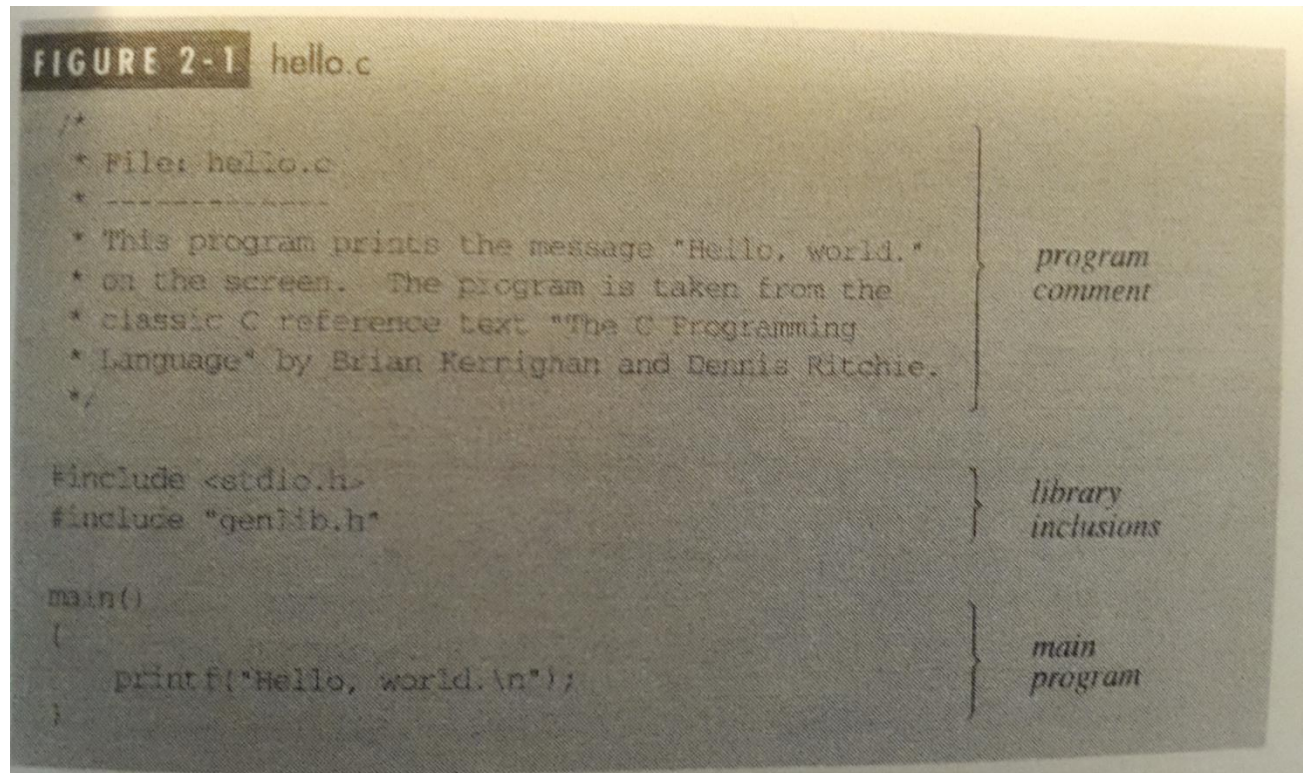
### **Programming project #1: Ch. 2: 1, 3, 4; Ch. 3: 5, 9**

Programs reproduced below for your convenience: (from the Programming Exercises section of each chapter)

## CH.2: (pg. 55-57)

1. filename: hello.c

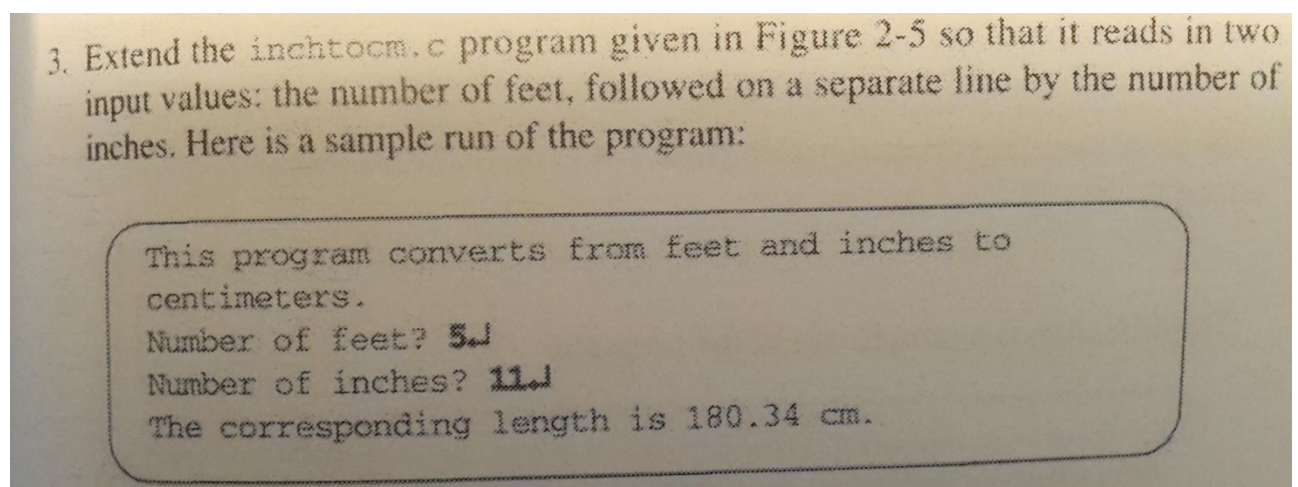
Type in the *hello.c* program exactly as it appears in this chapter and get it working.



*TA comment:* If the return type of main is void or not written (ie. void), there will be no "return 0".

*TA comment:* Make sure to use the pipe ">" command as shown above to send the output to a file instead of printing to the screen. Then "diff" the resulting files to check for differences.

3. filename: inchtocm.c






**FIGURE 2-5** inchtocm.c

```
/*
 * File: inchtocm.c
 * -----
 * This program reads in a length given in inches and converts it
 * to its metric equivalent in centimeters.
 */

#include <stdio.h>
#include "genlib.h"
#include "simpio.h"

main()
{
    double inch, cm;

    printf("This program converts inches to centimeters.\n");
    printf("Length in inches? ");
    inch = GetReal();
    cm = inch * 2.54;
    printf("%g in = %g cm\n", inch, cm);
}
```

*TA comment:* The  character means you press "Enter". You won't need this if you are testing with the input file.

*TA comment:* We have provided a sample input file as well as a sample output file for this program. Use diff to make sure the print statements match exactly with our given files. Use the inchtocm\_input file given with diff to check for a specific input(as shown in the notes above). Spaces do not matter as we will be using "diff -w" to ignore white spaces when checking.

#### 4. filename: interest.c

Write a program that reads in two numbers: an account balance and an annual interest rate expressed as a percentage. Your program should display the new balance after a year. There are no deposits or withdrawals - just the interest payment. Your program should be able to reproduce the following sample run:

```
Interest calculation program.  
Starting balance? 6000.1  
Annual interest rate percentage? 4.25.1  
Balance after one year: 6255
```

*TA comment:* Use %g to print all your numbers.

*TA comment:* We have provided a sample input file as well as a sample output file for this program.

### CH.3: (pg. 93-97)

5. filename: squares.c

5. Write a program that prints out the squares of the numbers from 1 to 10, using the format shown in the following sample run:

```
1 squared is 1  
2 squared is 4  
3 squared is 9  
4 squared is 16  
5 squared is 25  
6 squared is 36  
7 squared is 49  
8 squared is 64  
9 squared is 81  
10 squared is 100
```

Design your program so that the limits 1 and 10 are easy to change.

*TA comment:* Do not hardcode the print statements. Use a loop to iterate from 1 to 10. #define your min and max values.

*TA comment:* Use %d to print the numbers in printf.

9. filename: average.c

9. Using the `addlist.c` example as a model, write a program that reads in a list of integers until the user enters the value `-1` as a sentinel. At that point, the program should display the average of the values entered so far. Your program should be able to duplicate the following sample run:

```
This program averages a list of integers.  
Enter -1 to signal the end of the list.  
? 95  
? 100  
? 89  
? 91  
? 97  
? -1  
The average is 94.4
```

Writing this program requires more thought than writing the `addlist.c` program in the text and is a good test of your problem-solving abilities.

*TA comment:* We have provided a sample input file as well as a sample output file for this program.

*TA comment:* In C, 0 is set to be FALSE and 1 is set to be TRUE. So, you can do `while (1)` instead of `while(TRUE)`. Also, in the version of the C compiler in CSIF, TRUE is not defined by default. So, if you want to use TRUE, you will have to manually define it in the code. You can do this by saying: `#define TRUE 1`

*TA comment:* `addlist.c` is on the following page.

*TA comment:* Use `%g` to print your final average value.



**FIGURE 3-3** addlist.c

```
/*
 * File: addlist.c
 * -----
 * This program adds a list of numbers. The end of the
 * input is indicated by entering 0 as a sentinel value.
 */

#include <stdio.h>
#include "genlib.h"
#include "simpio.h"

main()
{
    int value, total;

    printf("This program adds a list of numbers.\n");
    printf("Signal end of list with a 0.\n");
    total = 0;
    while (TRUE) {
        printf(" ? ");
        value = GetInteger();
        if (value == 0) break;
        total += value;
    }
    printf("The total is %d\n", total);
}
```