Homework 3

Homework 3 is due on Wednesday, March 11, at the start of the lecture. Remember to include your *Discussions section* (e.g. ED1) and follow the complete Homework submission guidelines.

Please ask all questions about this assignment during the office hours, or post them on piazza.

Be sure to show your work on all your homework problems; otherwise, you risk not receiving credit.

It is recommended that you do not use a calculator (or online tools, etc.) when you do your homework. Remember that there will be no calculators allowed on exams.

Introduction to C programming

1. sample.c

Read and analyze the program below, then answer the following questions about the program's behavior when compiled and executed by a user.

1. If the user enters the following sequence of samples once per prompt, each followed by <Enter>—10, 20, -100, 42, 17, 19, 42, -104, 8, 5—what exactly does the program output after all samples have been entered?
2. If the user enters the following sequence of samples once per prompt, each followed by <Enter>—17, 19, 329, 0, 15, 42, 8, 4, 13, 5—what exactly does the program output after all samples have been entered?
3. What is the relationship between the samples entered by the user and the variable A printed at the end of the program? In other words, what value is printed for A in terms of the samples?
4. What is the relationship between the samples entered by the user and the variable B printed at the end of the program? In other words, what value is printed for B in terms of the samples?

sample.c

```c
#include <stdio.h>

int main ()
{
    int i;
    int sample;
    int A;
    int B;

    /* Read the first sample. */
    printf ("Enter the first sample: ");
    /* The expression "scanf (...)" returns the number of      *
     * values converted.  If the human user types a number     *
     * (as expected), the expression's value is 1.  Otherwise, *
     * the human did something wrong, so we end the program.   */
    if (1 != scanf ("%d", &sample)) {
        printf ("Numeric samples only!\n");
        /* Quit indicating failure (anything non-zero, *
         * by convention).                             */
        return 3;
    }

    /* Process the first sample. */
    A = sample;
    B = sample;

    for (i = 2; 10 >= i; i = i + 1) {
        /* Read another sample. */
        printf ("Enter sample #%d: ", i);
        if (1 != scanf ("%d", &sample)) {
            printf ("Numeric samples only!\n");
            /* Quit indicating failure (anything non-zero, *
             * by convention).                             */
            return 3;
        }

        /* Process the next sample. */
        if (A > sample) {
            A = sample;
        }
        if (B < sample) {
            B = sample;
        }
    }

    /* Print the results. */
    printf ("The A value is %d.\n", A);
    printf ("The B value is %d.\n", B);

    /* Program has finished successfully, *
     * so return 0 by convention.         */
    return 0;
}
```

2. average.c

Read and analyze the program below, then answer the following questions about the program’s behavior when compiled and executed by a user.

1. If the user enters the following sequence of samples once per prompt, each followed by <Enter>—10, 20, 30, 40, 50, 60, 70, 80, 90, 100—what exactly does the program output after all samples have been entered?
2. If the user enters the following sequence of samples once per prompt, each followed by <Enter>—19, 19, 19, 19, 19, 19, 19, 19, 19, 19—what exactly does the program output after all samples have been entered?
3. In one or two sentences, explain why the value printed by the program in response to the samples given in part (b) does not match the value that a human might expect. Use line numbers for specificity.
4. In one or two sentences, suggest a change to the program to make the calculation of the average more accurate.

average.c

```
#include <stdio.h>

int main ()
{
    int i;
    int sample;
    int average = 0;

    for (i = 1; 10 >= i; i = i + 1) {
        /* Read a sample. */
        printf ("Enter sample #%d: ", i);
        if (1 != scanf ("%d", &sample)) {
            printf ("Numeric samples only!\n");
            /* Quit indicating failure (anything non-zero, *
             * by convention).                             */
            return 3;
        }

        /* Process the next sample. */
        average = average + (sample / 10);
    }

    /* Print the results. */
    printf ("The average is %d.\n", average);

    /* Program has finished successfully, *
     * so return 0 by convention.         */
    return 0;
}
```

3. Binary arithmetic

Calculate the decimal value for each of the following C expressions, assuming that variable X has value 202, variable Y has value 42, and both have type **i nt** (assume 32-bit 2's complement).

1. (X & Y)
2. (X | Y)
3. (Y >> 3)
4. (X ^ Y)
5. (~X)
6. ((-Y) >> 3)
7. (X + Y) / 3

Section 1.5.4 of Lumetta's notes explains the meaning of C operators.

4. cryptic.c

Download, compile, and execute the program cryptic.c. Type in your name without spaces and record the output of the program after you have entered your name (do not include the prompt for input nor your typing). *Please note that you are not asked to understand the output nor to understand the program, which makes use of C constructs that you will learn in ECE220. We are only checking that you know how to compile and execute the program.*

5. Number of executions

Consider the C loop shown below. Variables x and i both have type **int**.

```
for (i = 0; x > i; i = i + 3) {
    /* This is the loop body. */
}
```

How many times does the loop body execute…

1. …when variable x is 20?
2. …when variable x is 42?
3. …when variable x is 16?
4. …when variable x is 0?

6. Conditional construct

The conditional construct

```
if (X >= 5) { printf ("Y"); }
```

is inserted at one of the “insertion points” in the code below.

```
if ( Y != 15 ) {
    printf ("X");
    /* INSERTION POINT A */
} else {
    /* INSERTION POINT B */
}
/* INSERTION POINT C */
```

For each insertion point (A, B, and C), indicate under what conditions—in other words, for what values of variables X and Y—the resulting code prints nothing, “X”, “Y”, and “XY.” For example, if the conditional construct based on X is NOT inserted, the answer should appear as follows:

The code

- prints “X” when Y 15,
- never prints “Y”. and
- never prints “XY”

7. Truth table

See Exams page. Solve problem #6 from UIUC **Spring 2016 Midterm 1** exam.