

## Homework 3: Basic C Programs

**1. Analysis of a C Program**

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

- a. If the user enters the following sequence of samples—17, -29, 33, 100, 50, -91, 217, 42, 42, 188—what exactly does the program output after all samples have been entered?

The A value is 217.

The B value is -91.

- b. If the user enters the following sequence of samples—0, 5, 9, 100, 99, 42, 18, 10, -10, 77—what exactly does the program output after all samples have been entered?

The A value is 100.

The B value is -10.

- c. 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?

A is the value of the largest sample (the maximum over samples).

- d. 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?

B is the value of the smallest sample (the minimum over samples).

```
#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. Analysis of a C Program, Part II

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

- a. If the user enters the following sequence of samples—80, 70, 50, 30, 40, 90, 10, 20, 60, 0—what exactly does the program output after all samples have been entered?

45

- b. If the user enters the following sequence of samples—8, 7, 5, 3, 4, 9, 1, 2, 6, 0—what exactly does the program output after all samples have been entered?

0

- c. 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.

The variable **average** is an integer; for all samples in part (b), (sample/10) evaluates to an integer value of 0. The sum of ten 0s is 0.

- d. In one or two sentences, suggest a change to the program to make the calculation of the average more accurate.

Sum up all inputs first, then divide the value by 10. Or use **float** instead of **int**.

```
#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. C Operators

Calculate the decimal value for each of the following C expressions, assuming that variable X has value 191, variable Y has value 27, and both have type `int` (assume 32-bit 2's complement).

For illustration, we'll use 9-bit 2's complement first. The higher 23 bits are copies of the sign bit.

- a.  $(X \& Y) = (0\ 1011\ 1111 \& 0\ 0001\ 1011) = 0\ 0001\ 1011 = 27_{10}$
- b.  $(X \mid Y) = (0\ 1011\ 1111 \mid 0\ 0001\ 1011) = 0\ 1011\ 1111 = 191_{10}$
- c.  $(Y \gg 3) = (0\ 0001\ 1011 \gg 3) = 0\ 0000\ 0011 = 3_{10}$
- d.  $(X \wedge Y) = (0\ 1011\ 1111 \wedge 0\ 0001\ 1011) = 0\ 1010\ 0100 = 164_{10}$
- e.  $(\sim X) = 1\ 0100\ 0000 = -192_{10}$
- f.  $((-Y) \gg 3)$  note: arithmetic right shift
  - i.  $(-Y) = \sim Y + 1 = 1\ 1110\ 0100 + 1 = 1\ 1110\ 0101$
  - ii.  $((-Y) \gg 3) = (1\ 1110\ 0101 \gg 3) = 1\ 1111\ 1100 = -4_{10}$
- g.  $(X + Y) / 3 = (0\ 1011\ 1111 + 0\ 0001\ 1011) / 3 = (0\ 1101\ 1010) / 3 = 218 / 3 = 72_{10}$

### 4. Finding Your Key

Download, compile, and execute the program `cryptic.c`. Type in your ZJUI ID number without spaces and record the output of the program. *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.*

### 5. Understanding Loops in C

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

```
for (i = 0; x > i; i = i + 4) {  
    /* This is the loop body. */  
}
```

How many times does the loop body execute...

- a. ...when variable x is 18? **5** (i takes values 0, 4, 8, 12, and 16 in the loop body)
- b. ...when variable x is 42? **11**
- c. ...when variable x is 99? **25**
- d. ...when variable x is 0? **0**

## 6. Understanding Conditionals in C

The conditional construct

```
if (-20 > Y) { printf ("Y"); }
```

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

```
if (42 == X) {  
    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 Y is NOT inserted, the answer should appear as follows:

The code

- prints nothing when  $X \neq 42$ ,
- prints “X” when  $X = 42$ ,
- never prints “Y,” and
- never prints “XY.”

At point A:

- prints nothing when  $X \neq 42$ ,
- prints “X” when  $X = 42$  and  $Y \geq -20$
- never prints “Y”
- prints “XY” when  $X = 42$  and  $Y < -20$

At point B:

- prints nothing when  $X \neq 42$  and  $Y \geq -20$
- prints “X” when  $X = 42$
- prints “Y” when  $X \neq 42$  and  $Y < -20$
- never prints “XY”

At point C:

- prints nothing when  $X \neq 42$  and  $Y \geq -20$
- prints “X” when  $X = 42$  and  $Y \geq -20$
- prints “Y” when  $X \neq 42$  and  $Y < -20$
- prints “XY” when  $X = 42$  and  $Y < -20$

## 7. Using Old Midterm Exams to Study

Do Problem #5 from ECE120 Fall 2016 Midterm #1. The exam is posted in the “Old Exams” portion of the class web page for you to use as a study tool.