16.216: ECE Application Programming

Fall 2011

Exam 1
October 5, 2011

Name: _____ ID #: _____ Section: _____

For this exam, you may use only one 8.5" x 11" double-sided page of notes. All electronic devices (e.g., calculators, cellular phones, PDAs) are prohibited. If you have a cellular phone, please turn it off prior to the start of the exam to avoid distracting other students.

The exam contains 3 questions for a total of 100 points. Please answer the questions in the spaces provided. If you need additional space, use the back of the page on which the question is written and clearly indicate that you have done so.

Please read each question carefully before you answer. In particular, note that:

- Question 3 has four parts, but you are only required to complete three of the four parts.
- The last page of the exam contains one or more sample runs of each program in Question 3—you may detach this page from the rest of the exam and do not have to submit it.
    - The output shown is input-dependent and does not necessarily test all input cases—do not assume that your program will always have that exact output!

You will have 50 minutes to complete this exam.

Q1: Multiple choice	/ 24
Q2: Reading code	/ 37
Q3: Writing code	/ 39
TOTAL SCORE	/ 100

1. (24 points, 6 points per part) ***Multiple choice***
For each of the multiple choice questions below, clearly indicate your response by circling or underlining the choice you think best answers the question.

a. Which of the following statements correctly defines `ClassNum` as a constant value?

i. `#constant ClassNum 16.216`

ii. `#define ClassNum 16.216`

iii. `#define ClassNum = 16.216`

iv. `double ClassNum = 16.216;`

b. Which of the statements below has the same result as `y = x * 2`?

i. `y = x >> 1;`

ii. `y = x << 1;`

iii. `y = x >> 2;`

iv. `y = x << 2;`

v. None of the above

1 (cont.)

c. Given an integer `val`, which of the following `if` statements prints `val` only if `val` is a positive, non-zero value that is less than 100?

i.
```
if ((val != 0) || (val < 100))
   printf("Value = %d\n", val);
```

ii.
```
if ((val != 0) && (val < 100))
   printf("Value = %d\n", val);
```

iii.
```
if ((val > 0) || (val < 100))
   printf("Value = %d\n", val);
```

iv.
```
if ((val > 0) && (val < 100))
   printf("Value = %d\n", val);
```

v. None of the above

d. Say you wish to print a double-precision variable, `d`, using precision `prec` and field width `width`. Which of the following statements correctly implements this operation?

i.
```
printf("%*lf", prec, d);
```

ii.
```
printf("%*lf", width, d);
```

iii.
```
printf("%10.5lf", d);
```

iv.
```
printf("%*.*lf", prec, width, d);
```

v.
```
printf("%*.*lf", width, prec, d);
```

2. (37 points) ***Reading code***

For each short program shown below, list the output exactly as it will appear on the screen. Be sure to clearly indicate spaces between characters when necessary.

You may use the available space to show your work as well as the output; just be sure to clearly mark where you show the output so that I can easily recognize your final answer.

a. (13 points)

```
int main() {
     int x, y;
     double d, e, f;

     x = 15;
     y = x + 5 / 4 - x;
     -x;

     d = 2;
     e = 10 / d;
     f = (x - 10.0) / 10;

     printf("x = %d, y = %d\n", x, y);
     printf("d = %lf,\ne = %lf,\nf = %lf\n", d, e, f);

     return 0;
}
```

2 (cont.)

b. (12 points)

```
int main() {
     float q, r, s;
     int p = 3, w = 9;

     q = 16.216;
     r = (q - 13.216) / 4 + 1.0;
     s = w / p * 10.0 + 0.6789;

     printf("q:%-10f", q);
     printf("r:%+*.0f", p, r);
     printf("s:%*.*f", w, p, s);

     return 0;
}
```

c. (12 points)

```
int main() {
     int x;
     printf("Enter value: ");
     scanf("%d", &x);
     switch (x - 2) {
     case 8:
     case 9:
          if ((x % 2) == 1)
               printf("Isn't that odd?\n");
          else
               printf("Let's make this even.\n");
          break;
     case -1:
          printf("That's less than I thought.\n");
     default:
          printf("How did we get here?\n");
     }
     return 0;
}
```

Complete the output for each of the following three possible inputs (input values underlined):

i. `Enter value: 11`

ii. `Enter value: 8`

iii. `Enter value: 1`

3. (39 points, 13 per part) ***Writing code***

For each part of this problem, you are given a short program to complete. **CHOOSE ANY THREE OF THE FOUR PARTS** and fill in the space provided with appropriate code. **If you complete all four, I will grade only the first three.**

a. Complete the program below so that it prompts for and reads three hexadecimal values, then prints each value in original form and with one additional change:

- First value: highest (bit 31) and lowest (bit 0) bits cleared (equal to 0)
- Second value: bit 16 flipped.
- Third value: middle eight bits (bits 12-19) set to 1

Outputs should contain 8 digits and a leading 0x. Leading zeroes should be added if necessary.

```
int main() {
     unsigned int hex1, hex2, hex3;     // Hexadecimal inputs









     return 0;
}
```

3 (cont.)

b. Complete the program below so that it prompts the user to enter the inputs shown, reads those values, and computes a weighted average, using the following variables:
   - `input1`, `input2`: The values to be averaged.
   - `percent1`, `percent2`: The weight for each value, written as a percentage.
     - `percent1 + percent2` should always equal 100.

The average should always be printed using three places after the decimal point.

```
int main() {
     double input1, input2;        // Values to be averaged
     int percent1, percent2;       // Weight for each value









     return 0;
}
```

3 (cont.)

c. Complete the program below so that it first prompts for and reads:
   - The upper and lower bounds of a range, written as `lower -> upper`
   - Two additional input values separated by a comma: `input1, input2`

The program should then print the following output:

- If `input1` is outside of the range `lower-upper`, print "`Input 1 out of range`".
- If the previous condition is false, but `input2` is inside the range `lower-upper` (including those end points), print "`Both inputs in range`".
- If neither of the previous conditions is true, print "`Conditions false`".

```
int main() {
     double lower, upper;          // Lower and upper bounds
     double input1, input2;        // Values to be checked
```

```
     return 0;
}
```

3 (cont.)

d. Complete the program below so that it prompts for and reads a single, case-insensitive character (i.e., treat `'A'` and `'a'` the same), followed by two integers. Depending on the character entered, your program should print the following information about these numbers:

- `'A'`, `'a'` → print the average of the integers in the form: `Average = <result>`
  - Always show one digit after the decimal point for the average.
- `'B'`, `'b'`, `'L'`, `'l'` → print the largest (most positive) of the integers in the form: `Largest = <result>`
- Any other character → print "`Invalid command`"

```
int main() {
     char cmd;           // Case-insensitive command input
     int num1, num2;     // Two numeric inputs









     return 0;
}
```

Sample outputs for programs in Question 3 (user inputs are underlined)

Part (a):

```
Hex values: 0xF162160F 0 12345678
0xF162160F 0x7162160E
0x00000000 0x00010000
0x12345678 0x123FF678
```

Part (b):

```
Inputs: 10 20
Weights: 25 75
Average: 16.875
```

Part (c):

```
Range: 2 -> 3
Inputs: 1.5, 2.5
Input 1 out of range

Range: 3.2 -> 10.7
Inputs: 4, 5
Both inputs in range

Range: 5.1 -> 10.2
Inputs: 7.3, 21.375
Conditions false
```

Part (d):

```
Command: A
Numbers: 3 2
Average: 2.5

Command: b
Numbers: 1 8
Largest: 8

Command: Q
Numbers: 83 -2
Invalid command
```