# 16.216: ECE Application Programming

Fall 2012

Exam 1 Solution

1. (20 points, 5 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. How many iterations does the loop below execute?

```
int x = 1;
while (x <= 20) {
   x = x * 2;
}
```

i. 2

ii. 4

***iii.*** ***<u>5</u>***

iv. 10

v. 20

b. Given the short code sequence below:

```
int z = 1;
do {
   printf("z = %d\n", z);
   z = z * -2;
} while (z < 0);
```

What will this program print?

i. Nothing

ii. `z = 1`

***iii.*** ***<u>`z = 1`</u>***
***<u>`z = -2`</u>***

iv.
```
z = 1
z = -2
z = 4
```

1 (cont.)

c. Given the code sequence below:

```
int x;
scanf("%d", &x);
switch (x / 4) {
case 0:
    printf("Zero ");
    break;
case 2:
    printf("Two ");
default:
    printf("Default");
}
```

Which of the following possible input values will produce the output “`Two Default`”?

A. 0
B. 2
C. 4
D. 8
E. 9

i. A and B

ii. Only B

iii. B and C

iv. Only D

***v.*** ***<u>D and E</u>***

d. Which of the following statements accurately reflect your opinion(s)? Circle all that apply (but please don’t waste too much time on this question)! ***<u>All answers accepted.</u>***

i. “This course is moving too quickly.”

ii. “This course is moving too slowly.”

iii. “I appreciate the opportunity to take a 50 minute nap three times a week.”

iv. “I hope the rest of the exam is this easy.”

v. “The programs get easier after Program 3, right? Please???”

2. (40 points) ***Expressions/operators***
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. (12 points)

```
#define Q2PartA 7 / 2.0

void main() {
     int intA, intB, intC;
     double doubA;

     intA = Q2PartA + 8;              intA = 7 / 2.0 + 8 = 3.5 + 8
                                           = 11.5 = 11 (truncated)

     intB = 15 – intA % 6;            intB = 15 – 11 % 6
                                           = 15 – 5 = 10

     intC = 30 / (intA + intB);       intC = 30 / (11 + 10)
                                           = 30 / 21 = 1.428571 …
                                           = 1 (truncated)

     doubA = intB – 0.5;              doubA = 10 – 0.5 = 9.5

     printf("Ints:\n%d %d %d", intA, intB, intC);
     printf("\nDouble: %lf", doubA);
}
```

***Output:***

```
     Ints:
     11 10 1
     Double: 9.500000
```

2 (cont.)

b. (14 points)

```
void main() {
     double x = 10;
     double y = 4;
     double out1, out2, out3;

     out1 = x / y / x;        out1 = 10 / 4 / 10
                                   = 2.5 / 10 = 0.25

     out2 = y * (y + 1);      out2 = 4 * (4 + 1) = 4 * 5 = 20

     out3 = 3.7 + 30 / x;     out3 = 3.7 + 30 / 10
                                   = 3.7 + 3 = 6.7

     printf("%5.2lf %+3.3lf\n", out1, out2);
          Print out1 with field width of 5, precision 2
          Print out2 with field width & precision of 3, and show
               sign of number even if positive

     printf("%-6.0lf\n", out3);
          Print out3 with field width of 6, precision of 0, and
               left justify value within field
}
```

**<u>Output:</u>** *Underlining used to show where spaces are; note that:*

- *On the first line, the value 0.25 has a single space before it, and there is a single space between the '5' in 0.25 and the '+' in +20.000*
- *On the second line, '7' is followed by 5 spaces.*

```
 0.25 +20.000
7     
```

c. (12 points)

```
void main() {
     unsigned int w, x, y;
     w = 0xFF0000 | 0xAA;      w = 0xFF00AA
     x = w ^ (0xBC << 8);      x = 0xFF00AA ^ 0xBC00 = 0xFFBCAA
     y = w & 0xF00F0;          y = 0xFF00AA & 0xF00F0 = 0xF00A0

     printf("%#.8X\n", w);     All values will print with a
     printf("%#.8X\n", x);     leading 0x and 8 digits shown
     printf("%#.8X\n", y);
}
```

***Output:***

```
     0x00FF00AA
     0x00FFBCAA
     0x000F00A0
```

3. (40 points, 20 per part) ***C input/output; conditional statements***
For each part of this problem, you are given a short program to complete. **CHOOSE ANY TWO OF THE THREE PARTS** and fill in the spaces provided with appropriate code. **You may complete all three parts for up to 10 points of extra credit, but must clearly indicate which part is the extra one—I will assume it is part (c) if you mark none of them.**

a. Complete the program below so that it prompts for and reads two integers representing a time in 24-hour format (0 ≤ hours ≤ 23), then prints the time in 12-hour format:

| **Valid range of hours (in 24 hr format)** | **AM / PM (12 hr format)** | **Earliest time (24 hr → 12 hr)** | **Latest time (24 hr → 12 hr)** |
|---|---|---|---|
| 0 ≤ hours ≤ 11 | AM | 0:00 == 12:00 AM | 11:59 == 11:59 AM |
| 12 ≤ hours ≤ 23 | PM | 12:00 == 12:00 PM | 23:59 == 11:59 PM |

All `printf()` calls are shown; you must fill in the blanks so they produce the right output. Examples are shown below, with user input underlined:

```
 Hrs/mins: 11 13          Hrs/mins: 14 5          Hrs/mins: 0 9
 11:13 AM                 2:05 PM                 12:09 AM
```

```
void main() {
     int hrs, mins;      // Time inputs: hours and minutes

     // Prompt for and read time in 24 hour format
     printf("Hrs/mins: ");
     scanf("%d %d", &hrs, &mins);

     // To handle "12:00" cases (hrs == 0 or 12),
     //    add 12 to hrs if hrs is divisible by 12
     if ((hrs % 12) == 0)
          hrs = hrs + 12;

     // Check if time is in morning and print accordingly
     if (hrs <= 12)
          printf("%d:%02d AM\n", hrs, mins);  // %.2d also works
                                               //    to format
                                               //    minutes

     // Otherwise, time must be in afternoon
     else
          printf("%d:%02d PM\n", hrs - 12, mins);

}
```

3 (cont.)

b. Complete the program below so that it first prompts for and reads two lines of input. The first contains the lower and upper bounds of a range; the second contains two values to be tested. The program should then print the following output:

- If `in2` is outside of the range, print "`I2 out of range`".
- If both are in the range (including the end points), print "`Both in range`".
- If neither of the previous conditions is true, print "`Conditions false`".

Three sample program runs are shown below, with user input underlined:

```
Range: 5 10             Range: 3.5 7.1          Range: 0 2.1
Values: 1 11            Values: 5.9 3.5         Values: 5 1
I2 out of range         Both in range           Conditions false
```

```
void main() {
     double low, hi;               // Lower and upper bounds
     double in1, in2;          // Values to be checked

     // Prompt for and read range
     printf("Range: ");
     scanf("%lf %lf", &low, &hi);

     // Prompt for and read values
     printf("Values: ");
     scanf("%lf %lf", &in1, &in2);

     // If in2 is out of range, print message below
     if ((in2 < low) || (in2 > hi))
          printf("I2 out of range\n");

     // If not, check if input1 is in range;
     //    if so, print message below
     else if ((in1 >= low) && (in1 <= hi))
          printf("Both inputs in range\n");

     // In all other cases, print message below
     else
          printf("Conditions false\n");
}
```

3 (cont.)

c. Complete the program below so that it prompts for and reads a single letter followed by two numbers. Depending on the character entered, your program should print the following information about these numbers, always showing one digit after the decimal point:

- `'A'`,`'M'` → print the average in the form: `Avg = <result>`
- `'D'` → print the difference `num1 – num2` in the form: `Diff = <result>`
- Any other character → print "`Invalid command`"

Note that all `printf()` calls are shown; you are responsible for filling in the blanks so that they produce the correct output.

Three sample program runs are shown below, with user input underlined:

```
Input: A 2 5          Input: D 0 4.27        Input: m 17 3
Avg = 3.5             Diff = -4.3            Invalid command
```

```
void main() {
     char cmd;              // Command input
     double num1, num2;     // Two numeric inputs

     // Prompt for and read input
     printf("Input: ");
     scanf("%c %lf %lf", &cmd, &num1, &num2);


     // Based on input, print average, max, or "Invalid command"
     switch (cmd) {
     case 'A':
     case 'M':
          printf("Avg = %.1lf\n", (num1 + num2) / 2);
          break;

     case 'D':
          printf("Diff = %.1lf\n", num1 – num2);
          break;
     default:
          printf("Invalid command\n");
     }
}
```