16.216: ECE Application Programming

Spring 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. Say you have the following code snippet:

```
int x = 5;
int y = 10;
if ((x % 2) == 1)
  y = y + x;
x = x + 1;
```

What are the values of `x` and `y` at the end of the sequence?

i. `x == 5, y == 10`

ii. `x == 6, y == 10`

iii. `x == 5, y == 15`

iv. ***`x == 6, y == 15`***

1 (cont.)

b. Say you have the following conditional statement:

```
if (a < 0) {
   x = x + 1;
}
else if (b >= 0) {
   x = x – 1;
}
else {
   x = 0;
}
```

Assume `x == 5` initially. Which values of `a` and `b` shown below would cause `x` to be set to 0 when the code above is executed?

i. `a == -2, b == -2`

ii. ***<u>`a == 1, b == -1`</u>***

iii. `a == 2, b == 5`

iv. `a == 0, b == 0`

c. Given an integer `val`, which of the following `if` statements prints `val` only if `val` is <u>not in the range</u> 0 through 100 (including the endpoints)?

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);
```

d. Which of the following `if` statements are completely valid? Circle all that apply.

i.
```
if (time_left_in_exam <= 5)
     panic = 1;
```

ii.
```
if (time_left_in_class_during_normal_lecture <= 25)
     printf("Zzzzzzzzzzzz ... \n");
```

iii.
```
if (I_submitted_a_regrade_request)
     I_am_wondering_where_my_new_grade_file_is = 1;
```

iv.
```
if (1) {
     printf("I just realized there’s no wrong answer ");
     printf("to this question.\n");
}
```

I think the answer is clearly (v):
```
if (1) {
     printf("Why can’t all the questions be this easy?\n");
}
```

2. (40 points) ***Expressions and 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. (15 points)

```
#define ConstVal 2.5

void main() {
     int ivarA, ivarB;
     double dvarA, dvarB;

     ivarA = ConstVal + 3;
     ivarB = 5 / (ivarA – 2) + 3;

     dvarA = 3 * ConstVal;
     dvarB = -ivarA / 2 + 2;

     printf("Integers: %d\n%d\n", ivarA, ivarB);
     printf("\nDoubles: %lf %lf\n", dvarA, dvarB);
}
```

Solution:

```
ivarA = 2.5 + 3 = 5.5 = 5 (result truncated since ivarA is int)
ivarB = 5 / (5 – 2) + 3 = 5 / 3 + 3 = 1 + 3 = 4
dvarA = 3 * 2.5 = 7.5
dvarB = -5 / 2 + 2 = -2 + 2 = 0

Actual output:
Integers: 5
4

Doubles: 7.500000 0.000000
```

2 (cont.)

b. (13 points)

```
void main() {
     float f1, f2, f3;
     int x;

     f1 = 2 + 24.0 / 100;
     f2 = (f1 + 1.76) / 4 * 0.125;
     f3 = f1 / 2 + 0.8 / 2;
     x = f1;

     printf("f1 = %f\n", f1);
     printf("f2 = %.2f\n", f2);
     printf("f3 = %.*f\n", x, f3);
}
```

Solution

```
f1 = 2 + 24.0 / 100 = 2 + 0.24 = 2.24
f2 = (2.24 + 1.76) / 4 * 0.125 = 4.0 / 4 * 0.125 = 0.125
f3 = 2.24 / 2 + 0.8 / 2 = 1.12 + 0.4 = 1.52
x = 2.24 = 2 (truncated since x is int)

Actual output:
f1 = 2.240000
f2 = 0.13
f3 = 1.52
```

c. (12 points)

```
void main() {
     unsigned int H1, H2, H3;

     H1 = 0x101 | 0x404;
     H2 = ~H1 >> 8;
     H3 = H1 ^ 0xF0F;

     printf("H1: %x\n", H1);
     printf("H2: %x\n", H2);
     printf("H3: %x\n", H3);
}
```

Solution:

H1 = 0x101 | 0x404= $0001\ 0000\ 0001_2$ OR
$0100\ 0000\ 0100_2$
= $0101\ 0000\ 0101_2$ = 0x505

H2 = ~(0x505) >> 8 = 0xFFFFFAFA >> 8 = 0x00FFFFFA

H3 = 0x505 ^ 0xF0F= $0101\ 0000\ 0101_2$ XOR
$1111\ 0000\ 1111_2$
= $1010\ 0000\ 1010_2$ = 0xA0A

Actual output:

```
H1: 505
H2: fffffa
H3: a0a
```

Remember: %x prints letters in lowercase--%X gives you uppercase
Leading 0s aren't printed unless you use the 0 flag
Leading 0x isn't printed unless you use the # flag

3. (40 points, 20 per part) ***C input/output***
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 space 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 three unsigned decimal integers, then reprints them on three lines in hexadecimal using the following formatting:

- First value: show the value with a leading 0x but no other formatting.
- Second value: show exactly the lowest 16 bits of the value with a leading 0x. All 16 bits must be shown, including extra zeroes if necessary.
- Third value: show all 32 bits of the value with a leading 0x and extra zeroes if necessary.

Examples (with user input underlined; in the second case, note that 65792_{10} = 0x10100, but only the lowest 16 bits are shown):

```
Enter three values: 10 32 500     Enter three values: 500 65792 3
0xA                               0x1F4
0x0020                            0x0100
0x000001F4                        0x00000003
```

Students had to complete the underlined, boldfaced lines

```
void main() {
     unsigned int in1, in2, in3;   // Hexadecimal inputs

     // Prompt for and read three unsigned decimal values
     printf("Enter three values: ");
     scanf("%u %u %u", &in1, &in2, &in3);

     // Print the hexadecimal form of the first value
     printf("%#.0x\n", in1);  // Accepted %#x; precision of 0
                              //    necessary for in1 == 0
                              //    to print correctly

     // Print the lowest 16 bits of the second value
     printf("%#.4x\n", in2 & 0x0000FFFF);
     // Also accepted: %#06x (see above)

     // Print all 32 bits of the third value
     printf("%#.8x\n", in3);
     // Also accepted: %#010x (see above)

}
```

3 (cont.)

b. Complete the program below so that it prompts the user to enter four input values—the initial balance in a bank account, followed by the month, day, and amount of the most recent transaction—then prints two lines of output:

- The initial account balance, which is between 0.00 and 10000.00
- The transaction date and amount, as well as the account balance after the transaction
  - The date should be printed in MM/DD format (examples: 02/24, 12/03, 10/15)
  - The transaction amount is between -9999.99 and +9999.99, and its sign should always be shown.

Both the balance and transaction amount should be shown with two places after the decimal point. Each should always line up in the same way when printed, regardless of the number of digits. Examples are shown below, with user input underlined:

```
Input balance, month, day, and amount: 500 2 24 250
Balance:   500.00
02/24  +250.00   750.00

Input balance, month, day, and amount: 10.50 12 5 -50.25
Balance:    10.50
12/05   -50.25   -39.75
```

Students had to complete the underlined, boldfaced lines

```
void main() {
     double bal, amt;    // Balance and transaction amount
     int mth, day;       // Month and day of transaction

     // Prompt for and read balance, month, day, and amount
     printf("Input balance, month, day, and amount: ");
     scanf("%lf %d %d %lf", &bal, &mth, &day, &amt);

     // Print initial balance
     printf("Balance: %8.2lf\n", bal);

     // Print date (MM/DD format), amount, and balance
     printf("%02d/%02d %+8.2lf %8.2lf\n",
               mth, day, amt, bal+amt);
}
```

3 (cont.)

c. Complete the program below so that it prompts the user to enter a single integer followed by two double-precision variables, reads those values, and does the following:

- Reprint each double-precision value, along with the percentage of their total each value comprises (See the test cases below).
    - The integer, p, determines how many places should be printed for each value
    - To print a percent sign, use `%%` in your format string
        - For example, if `x == 5`, `printf("%d%%", x)` will output `5%`

```
Int + 2 doubles: 1 2 3
Input 1: 2.0 40.0%
Input 2: 3.0 60.0%
```

```
Int + 2 doubles: 3 0.1 0.3
Input 1: 0.100 25.000%
Input 2: 0.300 75.000%
```

Students had to complete the underlined, boldfaced lines

```
void main() {
     int p;                                            // Output precision
     double input1, input2;                            // Input values

     // Prompt for and read integer + 2 doubles
     printf("Int + 2 doubles: ");
     scanf("%d %lf %lf", &p, &input1, &input2);

     /* Print each of the two doubles, followed by the
          percentage of their total each value comprises */
     printf("Input 1: %.*lf %.*lf%%\n", p, input1, p,
               input1 / (input1 + input2) * 100);
     printf("Input 2: %.*lf %.*lf%%\n", p, input2, p,
          input2 / (input1 + input2) * 100);
}
```