16.216: ECE Application Programming

Summer 2012

Exam 1 Solution

1. (20 points, 5 points per part) ***<u>Multiple choice</u>***

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:

```
double v1, v2;
scanf("%lf %lf", &v1, &v2);
if ((v1 < v2) && (v1 > 5.0))
  printf("Condition true\n");
```

Which of the possible lines of input below will cause this code to print `Condition true`?

i. `0.75 3.75`

ii. `45 5.4`

iii. `5.0 6.7`

iv. ***<u>`5.1 6.7`</u>***

v. `Condition true`

b. Say you have the following conditional statement (assume all variables are of type `int`):

```
if (b + c == 3) {
  a = 5;
}
else if (b < c) {
  a = 17;
}
else {
  a = 34;
}
```

Which values of `b` and `c` shown below would cause `a` to be set to 17 when the code above is executed?

i. `b == 2, c == 2`

ii. `b == -2, c == 5`

iii. ***`b == 2, c == 5`***

iv. `b == 12345, c == 0`

c. Say you have the following code snippet:

```
scanf("%d", &x);
switch(x % 3) {
case 0:
  printf("Result 0 ");
case 1:
  printf("Result 1 ");
  break;
default:
  printf("Default");
}
```

If you run the program containing this code twice, and input the values 9 and 8, respectively, what will the outputs be?

i. First run (x == 9): `Result 0`
   Second run (x == 8): `Default`

ii. <u>First run (x == 9): `Result 0 Result 1`</u>
   <u>Second run (x == 8): `Default`</u>

iii. First run (x == 9): `Result 0 Result 1 Default`
   Second run (x == 8): `Default`

iv. First run (x == 9): `Default`
   Second run (x == 8): `Default`

v. First run (x == 9): `Default`
   Second run (x == 8): `Result 0 Result 1`

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)! ***(All answers are "right")***

i. "I'd rather be outside than stuck in class."

ii. "This course is moving too quickly."

iii. "This course is moving too slowly."

iv. "I hope the rest of the exam is this easy."

v. "We don't really have to show up for the 'extra' Friday classes, do we?"

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. (12 points)

```
#define Q2PartA 3.0 / 2

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

     intA = 7 + Q2PartA;
     intB = 2 + intA / 3 * 2;
     intC = 3 * (intA - intB);

     doubA = intA + Q2PartA;

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

Solution:

```
  intA = 7 + Q2PartA = 7 + 3.0 / 2 = 7 + 1.5 = 8 (int truncates)

  intB    = 2 + intA / 3 * 2
          = 2 + 8 / 3 * 2 = 2 + 2 * 2 = 2 + 4 = 6

  intC = 3 * (intA - intB) = 3 * (8 – 6) = 3 * 2 = 6

  doubA = intA + Q2PartA = 8 + 3.0 / 2 = 8 + 1.5 = 9.5
```

OUTPUT:

```
  Integers: 8 6
  6Double:
   9.500000
```

2 (cont.)

b. (14 points)

```
void main() {
     double d1, d2, d3;
     int x;

     x = 7;
     d1 = x + 3.5 / 5;
     d2 = (d1 + 2.3) / 4 + 0.008;
     d3 = x / 2.0 + 1.1;
     x = d2;

     printf("d1 = %-10.3f\n", d1);
     printf("d2 = %*.*f\n", x, x, d2);
     printf("d3 = %+.0f\n", d3);
}
```

Solution:

```
  d1 = x + 3.5 / 5 = 7 + 3.5 / 5 = 7 + 0.7 = 7.7

  d2 = (d1 + 2.3) / 4 + 0.008
     = (7.7 + 2.3) / 4 + 0.008 = 10.0 / 4 + 0.008 = 2.5 + 0.008
     = 2.508

  d3 = x / 2.0 + 1.1 = 7 / 2.0 + 1.1 = 3.5 + 1.1 = 4.6

  x = d2 = 2 (int truncates)
```

OUTPUT:

```
  d1 = 7.700                      (NOTE: 5 spaces after last 0)
  d2 = 2.51
  d3 = +5
```

c. (14 points)

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

     H1 = 0xFF0 & 0x123;
     H2 = ~(H1 << 8);
     H3 = H1 ^ 0x0000FF00;

     printf("H1: %x\n", H1);
     printf("H2: %#x\n", H2);
     printf("H3: %#.8x\n", H3);
}
```

Solution:

```
  H1 = 0xFF0 & 0x123 = 0x120 (0x00000120 as a full 32-bit value)
  H2 = ~(H1 << 8) = ~(0x120 << 8) = ~0x12000 = 0xFFFEDFFF
  H3 = 0x120 ^ 0x0000FF00 = 0x0000FE20
```

OUTPUT:

```
  H1: 120
  H2: 0xfffedfff
  H3: 0x0000fe20
```

NOTE: I'll also accept 0x00fe20 as the output value for H3, as there's a typo in the lecture slides that says that, when used with hexadecimal output, the precision specifies the number of characters to be printed--including the leading "0x". In fact, the precision specifies the number of digits to be printed and does not include the leading 0x.

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 this program so that it prompts for and reads three hexadecimal values (`in1`, `in2`, `in3`) and a single integer (`x`), then prints the following on three separate lines:

- First line: show `in1` both in decimal and in hexadecimal, with the hexadecimal output showing all 32 bits with a leading 0x and extra zeroes if necessary.
- Second line: show `in2` multiplied by 2^x, in hexadecimal. This output should have only a leading 0x, with no additional formatting.
- Third line: show the lowest 12 bits of `in3`, with a leading 0x and extra zeroes if necessary.

Examples (with user input underlined):

```
3 hex, 1 int: A 80 11F3 2          3 hex, 1 int: 0100 0xF 0x3 10
10 0x0000000A                      256 0x00000100
0x200                              0x3C00
0x1F3                              0x003
```

Code to be added is bold, italic, underlined

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

     // Prompt for and read three hexadecimal values and one int
     printf("3 hex, 1 int: ");
     scanf("%x %x %x %d", &in1, &in2, &in3, &x);

     // Print in1 in decimal and hexadecimal (show all 32 bits)
     printf("%d %#.8X\n", in1, in1);     // NOTE: will accept
                                         //  %#.10x as second
                                         //  format specifier,
                                         //  due to lecture typo
                                         //  discussed above

     // Print in2 multiplied by 2x
     printf("%#X\n", in2 << x);

     // Print only the lowest 12 bits of in3
     printf("%#.3X\n", in3 & 0x0FFF);    // NOTE: will accept
                                         //  %#.5x, for reasons
                                         //  cited above
}
```

3 (cont.)

b. Complete this program so that it prompts the user to enter four input values representing a baseball player's statistics—hits (H), walks (BB), total bases (TB), and at bats (AB), then calculates and prints the following values, each on its own line:

- Batting average (AVG): the fraction of at bats in which the player gets a hit
- On-base percentage (OBP): the fraction of plate appearances (PA = AB + BB) in which the player reaches base by getting either a walk or a hit.
- Slugging percentage (SLG): the ratio of total bases to at bats.
- OPS: the sum of OBP and SLG.

Each value should be printed using 3 decimal places. Examples are shown below, with user input underlined:

```
Enter H / BB / TB / AB: 100 50 180 450
AVG: 0.222
OBP: 0.300
SLG: 0.400
OPS: 0.700
```

Code to be added is bold, italic, underlined

```
void main() {
     double H, BB, TB, AB, PA;     // Stats described above

     // Prompt for and read hits, walks, total bases, at bats
     printf("Enter H / BB / TB / AB: ");
     scanf("%lf %lf %lf %lf", &H, &BB, &TB, &AB);

     // Calculate PA
     PA = AB + BB;

     // Print AVG, OBP, SLG, and OPS
     printf("AVG: %.3lf\n", H / AB);
     printf("OBP: %.3lf\n", (H + BB) / PA);
     printf("SLG: %.3lf\n", TB / AB);
     printf("OPS: %.3lf\n", ((H + BB) / PA) + (TB / AB));
}
```

3 (cont.)

c. Complete this program so that it prompts the user to enter a student ID number followed by three more non-negative integers, reads those values, and prints the following on one line:

- The student ID number, as an 8-digit value.
    - If the ID has less than 8 digits, leading zeroes should fill the remaining places.
- Each of the three integers, with at least one space between each
    - The integers should always line up the same way, regardless of the number of digits
    - Each integer has a minimum value of 0 and a maximum value of 100.
- The average of the three integers, always shown with 2 digits after the decimal point

```
Enter ID and 3 ints: 12345678 50 60 70
12345678  50  60  70 60.00

Enter ID and 3 ints: 33 5 100 35
00000033   5 100  35 46.67
```

Code to be added is bold, italic, underlined

```
void main() {
     unsigned int ID;                // Student ID
     unsigned int v1, v2, v3;        // Three integer inputs

     // Prompt for and read ID number + 3 integers
     printf("Enter ID and 3 ints: ");
     scanf("%u %u %u %u", &ID, &v1, &v2, &v3);

     /* Print 8-digit ID, 3 integer values (always aligned the
        same way), & average (with 2 digits after decimal pt) */
     printf("%08u %3u %3u %3u %.2lf\n",
                ID, v1, v2, v3, (v1 + v2 + v3) / 3.0);
}
```