# 16.216: ECE Application Programming

Fall 2015

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 one choice you think best answers the question.

a. What is the output of the short code sequence below?

```
int i = 10;
while (i > 0) {
   i = i / 2;
   printf("%d ", i);
}
```

i. `10`

ii. `5`

iii. `10 5 2 1`

***iv. 5 2 1 0***

v. `10 5 2 1 0`

b. What is the output of the short code sequence below?

```
int z = -3;
do {
   printf("%d ", z);
   z = z * -2;
} while (z > 0);
printf("%d ", z);
```

i. No output—loop body never executes

ii. `-3`

iii. `-3 6`

iv. `-3 6 6`

***v. -3 6 -12***

1 (continued)

c. Given the code sequence below:

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

Which of the following possible input values will produce the output "`Zero Two `"?

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

i. Only A

***ii. <u>A and B</u>***

iii. A and C

iv. B and E

v. A, C, and D

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")!

i. "This course is moving too quickly."

ii. "This course is moving too slowly."

iii. "I've attended very few lectures, so I don't really know what the pace of the course is."

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

***<u>Any of the above are "correct."</u>***

2. (40 points) ***C input/output; 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)

```
int main() {
    int v1, v2;
    double d1;
    double d2 = 18;

    v1 = d2 / 5.0;        v1 = 18 / 5.0 = 3.6 = 3 (truncate to int)
    v2 = v1 + 15 / v1;    v2 = 3 + 15 / 3 = 3 + 5 = 8
    d1 = d2 * 0.5 - v1;   d1 = 18 * 0.5 – 3 = 9 – 3 = 6
    d2 = v2 / v1;         d2 = 8 / 3 = 2 (int division)

    printf("%d\n%d ", v1, v2);
    printf("\n%lf %lf\n", d1, d2);

    return 0;
}
```

***Note that d1 and d2 print with 6 digits shown after the decimal point, as no alternative precision is specified for these values.***

**OUTPUT:**

```
3
8
6.000000 2.000000
```

2 (continued)
b. (14 points)

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

    d1 = 11.0 / 4;          d1 = 11.0 / 4 = 2.75
    x = d1 + 10.25;         x = 2.75 + 10.25 = 13 (conversion to
                            int doesn't happen until after addition
                            is done)
    d2 = -(d1 * 2) + x;     d2 = -(2.75 * 2) + 13 = -5.5 + 13 = 7.5
    d3 = d2 / 100;          d3 = 7.5 / 100 = 0.075

    printf("%.0lf\n", d1);
    printf("%.2lf\n", d2);
    printf("%.1lf\n", d3);    // Print d3 with a precision of 1
    printf("%d\n", x);

    return 0;
}
```

***Note that, in the output, d1 is rounded to 3, the nearest integer (precision of 0); d2 is printed as 7.50 (precision of 2), and d3 is rounded to 0.1 (precision of 1).***

**OUTPUT:**

```
3
7.50
0.1
13
```

2 (continued)
c. (14 points)

For this program, assume the user inputs the line below. The digit `'9'` is the first character the user types. There are two spaces (`' '`) between the `'5'` in `9.30.2015` and the `'9'` in `900`, and two spaces between the second `'0'` in `900` and the `'1'` in `1000`.

You must determine how `scanf()` handles this input and then print the appropriate results. Note that the program may not read all characters on the input line.

```
  9.30.2015  900  1000

int main() {
   int ival1, ival2;
   double dval1, dval2;
   char ch1, ch2, ch3;

   scanf("%d%c%lf %c %c %d %lf",
          &ival1, &ch1, &dval1, &ch2,
          &ch3, &ival2, &dval2);

   printf("%d %d\n", ival1, ival2);
   printf("%.4lf %.4lf\n", dval1, dval2);
   printf("%c%c%c\n", ch1, ch2, ch3);

   return 0;
}
```

***This program reads its input as follows:***

- ***ival1 holds the first integer:*** ***ival1 = 9***
- ***ch1 holds the next character:*** ***ch1 = '.'***
- ***dval1 holds the next number:*** ***dval1 = 30.2015***
- ***ch2 holds the next non-space character:*** ***ch2 = '9'***
- ***ch3 holds the next non-space character:*** ***ch3 = '0'***
- ***ival2 holds the next integer:*** ***ival2 = 0***
- ***dval2 holds the next number:*** ***dval2 = 1000***

**OUTPUT:**

```
9 0
30.2015 1000.0000
.90
```

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.**

Remember, you must write all code required to make each program work as described—**you cannot simply fill in the blank lines and get full credit.** Also, remember that each example only applies to one specific case—**it does not cover all possible results for that program.**

a. This program reads three values representing the sides of a triangle. If any of the values are negative, the program prints an error and exits. Otherwise, it prints the type of triangle:

- If all three sides are equal, the triangle is equilateral.
- If two of the three sides are equal, the triangle is isosceles.
- If none of the sides are equal, the triangle is scalene.

Three test runs of the program are shown below, with user input underlined.

```
Sides: 1.2 3.4 1.2      Sides: 9.8 15 7         Sides: 5 -2 3
Isosceles               Scalene                 Error: negative side
```

```
void main() {
   double s1, s2, s3;      // Sides of a triangle
   // Prompt for and read triangle sides
   printf("Sides: ");
   scanf("%lf %lf %lf", &s1, &s2, &s3);

   // Input error: negative side length
   if ((s1 < 0) || (s2 < 0) || (s3 < 0)) {
      printf("Error: Negative side\n");
      return;
   }

   // Determine triangle type--write code around printf() calls
   if ((s1 == s2) && (s2 == s3))
      printf("Equilateral\n");

   else if (s1 == s2 || s1 == s3 || s2 == s3)
      printf("Isosceles\n");

   else
      printf("Scalene\n");
}
```

3 (continued)

b. In the card game blackjack, players use a standard strategy to determine when to "hit" (take another card), "stand" (take no more cards), or "double down" (double your bet and hit). A simplified (and slightly incorrect) version of that strategy, which is based on the player's total and the one dealer card showing, is as follows:

   - The player doubles down if his total is 10 or 11 and the dealer's card is less than 10.
   - The player stands if his total is 17 or higher, or if his total is between 12 and 16 and the dealer's card is 6 or less.
   - The player hits in all other cases.

Complete the program below, which reads two integers (the player's total and the value of the dealer's card) and prints the correct choice based on the conditions described above.

Three sample program runs are shown below (user input underlined):

```
Player/dealer: 17 10     Player/dealer: 13 8      Player/dealer: 10 9
Stand                    Hit                      Double down
```

```
void main() {
   int plr;     // Player total
   int dlr;     // Value of dealer card

   // Prompt for and read player total and dealer card
   printf("Player/dealer: ");
   scanf("%d %d", &plr, &dlr);

   // Test conditions to determine when to double, hit, or stand
   // Write code around printf() calls
   if (dlr < 10 && (plr == 10 || plr == 11))
      printf("Double down\n");

   else if (plr >= 17 || (plr >= 12 && plr <= 16 && dlr <= 6))
      printf("Stand\n");

   else
      printf("Hit\n");

}
```

3 (continued)

c. This program prompts the user to enter a single character command followed by three numbers: two integers representing the width and length of a rectangle, and a third integer representing the precision of the printed result. The command determines how the result is calculated:

- If the command is 'P' or 'p,' calculate the rectangle's perimeter (distance around the outside of the rectangle).
- If the command is 'A' or 'a', calculate the rectangle's area (product of its sides).
- For all other characters, print an error message showing the invalid command and exit.

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

```
Enter char, 3 ints: A 2 4 5          Enter char, 3 ints: P 4 8 0
Area = 8.00000                       Perimeter = 32 24
```

```
void main() {
   int width, length;     // Width and length of rectangle
   int prec;              // Precision of output
   char cmd;              // Single character command

   // Prompt for/read input values
   printf("Enter char, 3 ints: ");
   scanf("%c %d %d %d", &cmd, &width, &length, &prec);

   // Test cmd to determine what value to print
   // Complete printf() calls and write code needed around them
   switch (cmd) {
   case 'A':
   case 'a':
      printf("Area = %.*lf\n", prec, width * length);
      break;

   case 'P':
   case 'p':
      printf("Perimeter = %.*lf\n", prec, 2 * (width + length));
      break;
      // Also accepted width * length for perimeter formula,
      //   given the typo in the perimeter test case

   default:
      printf("Invalid command %c\n", cmd);
}
```