16.216: ECE Application Programming

Fall 2014

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?

```
x = 1;
while ((x % 3) != 0) {
   printf("%d ", x);
   x = x + 4;
}
```

i. No output

ii. `1`

iii. ***`1 5`***

iv. `1 5 9`

v. `1 5 9 13`

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

```
int x = 3;
int y = -3;
do {
   printf("* ");
   y = y + 1;
   x = -y + 2;
} while ((x > y) && (x > 2));
```

i. No output

ii. *

iii. * *

iv. **** * ****

v. * * * *

c. Given the following code snippet:

```
int x = 100;
for (i = 1; i < 5; i++) {
   x = x – 10;
}
```

Which of the following choices can replace the `for` loop and produce the exact same value for `x`? Assume `x` is always initialized to 100.

i.
```
i = 0;
while (i < 8) {
   x += (-10);
   i += 2;
}
```

ii.
```
i = 5;
while (i >= 1) {
   x = x – 10;
   i--;
}
```

iii.
```
i = 0;
while (i < 5) {
   x = x – 10;
   i++;
}
```

iv.
```
i = x;
while (i > 60) {
   i = i – 10;
}
```

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 this easy."

Any of the above are "correct."

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)

```
void main() {
   double d1, d2;
   int v1;
   int v2 = 16;

   d1 = v2 / 3;                d1 = 16 / 3 = 5 (int division)
   d2 = (v2 + 1.0) / 2;        d2 = (16 + 1.0) / 2 = 17.0 / 2
                                   = 8.5
   v1 = d1 * 2 + 7;            v1 = 5 * 2 + 7 = 10 + 7 = 17
   v2 = v2 % v1;               v2 = 16 % 17 = 16

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

OUTPUT

```
5.000000 8.500000 17
16
```

2 (continued)
b. (14 points)

```
void main() {
   int in1;
   float f1, f2;
   double do1;

   in1 = 20;
   f1 = in1 / 40.0 + 3;
   f2 = f1 + in1;
   do1 = (f2 - f1) / (in1 - 15);
   in1 = f1 + f2;

   printf("%06d\n", in1);
   printf("%-6.2f %+6.2f\n", f1, f2);
   printf("%10.4lf\n", do1);
}
```

f1 = 20 / 40.0 + 3 = 0.5 + 3 = 3.5

f2 = 3.5 + 20 = 23.5

do1 = (23.5 – 3.5)/(20 – 15) = 20.0 / 5 = 4.0

in1 = 3.5 + 23.5 = 27 (int)

OUTPUT: *Note that:*

- *There are three spaces between 3.50 and +23.50. The first two are part of the field used to print 3.50 with a field width of 6. The third space is the space between format specifiers.*
- *There are four spaces before the number 4.0000, since that value is printed with a field width of 10.*

```
000027
3.50   +23.50
    4.0000
```

2 (continued)

c. (14 points)

For this program, assume the user inputs the line below. The digit `'1'` is the first character the user types. The plus sign (`'+'`) is followed by a single space character (`' '`), while the digits `'4'` and `'5'` are separated by two spaces.

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.

```
  110+ 0.1234  5

void main() {
   int ival;
   double dval;
   char ch1, ch2, ch3, ch4;

   scanf("%c%d%c %c %lf %c",
      &ch1, &ival, &ch2, &ch3, &dval, &ch4);

   printf("%d %.4lf\n", ival, dval);
   printf("%c%c%c%c\n", ch1, ch2, ch3, ch4);
}
```

This program reads its input as follows:

- *`ch1` is the first character* → *`ch1 = '1'`*
- *`ival` holds the next whole number* → *`ival = 10`*
- *`ch2` is the first character after the 0 in "110"* → *`ch2 = '+'`*
- *`ch3` is the first non-space character after '+'* → *`ch3 = '0'`*
- *`dval` holds the next number* → *`dval = .1234`*
- *`ch4` is the first non-space character after `0.1234`* → *`ch4 = '5'`*

OUTPUT:

```
10 0.1234
1+05
```

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. The first three digits of a 5-digit zip code are based on the state in which the zip code is located. For example, all Massachusetts zip codes start with digits between 010 and 027, and all New Hampshire zip codes start with digits between 030 and 038.

   Your program should read a zip code and determine if it is in one of these two states. The program should also check that the input is properly formatted as an integer, printing an error message if not. Three test runs of the program are shown below, with user input underlined.

```
Enter zip: 01854        Enter zip: 06320        Enter zip: X6320
01854 --> MA            06320 outside MA & NH   Input error
```

Note that you can read and print integers with leading zeroes, but you should not use leading zeroes for constant values in your program. For example, to check if the user entered the university's zip code, you would compare `zip` to 1854, not 01854. Note also that leading zeroes do not affect how you read an integer, but you will need extra formatting to print leading zeroes.

Students were responsible for the underlined, boldfaced, italicized code.

```
void main() {
   int zip;          // Input zip code
   int n;            // Test for input error

   // Prompt for and read zip code
   printf("Enter zip: ");
   n = scanf("%d", &zip);

   if (n < 1)                  // Test for input error
      printf("Input error\n");

   else if ((1000 <= zip) && (zip <= 2799))  // MA zip codes
      printf("%05d --> MA\n", zip);

   else if ((3000 <= zip) && (zip <= 3899))  // NH zip codes
      printf("%05d --> NH\n", zip);

   else
      printf("%05d outside MA & NH\n", zip);
}
```

3 (continued)

b. Given three double-precision input values, this program should print the greatest of the inputs using a precision of 2. Two sample program runs are shown below (user input underlined):

```
Enter three values: 1.2 3.4 5.6
5.60 is max value

Enter three values: -7 3 1
3.00 is max value
```

Students were responsible for the underlined, boldfaced, italicized code.

```
void main() {
   double d1, d2, d3;          // Three inputs
   double max;                 // Maximum value

   // Prompt for and read three input values
   printf("Enter three values: ");
   scanf("%lf %lf %lf", &d1, &d2, &d3);

   // Use if/else statements to find maximum value. HINT: Use
   //  nested if statements—compare two inputs (which will help
   //  you eliminate one) with an if statement, then use an
   //  if/else statement inside that block to choose one of the
   //  other inputs. Handle the outer else case similarly.
   if (d1 > d2) {          // d1 or d3 max
      if (d1 > d3)
         max = d1;
      else
         max = d3;
   }

   else {                  // d2 or d3 max
      if (d2 > d3)
         max = d2;
      else
         max = d3;
   }

   // Print maximum value
   printf("%.2lf is max value\n", max);
}
```

3 (continued)

c. This program prompts for and reads a double-precision value followed by a single character representing one of three prefixes: 'k', for kilo- (10^3 = 1 thousand), 'M', for mega- (10^6 = 1 million), or 'G', for giga- (10^9 = 1 billion). Depending on the character, your program should multiply the input value by the appropriate amount and show the full value rounded to the nearest integer. If the user enters an invalid character, print an appropriate error message.

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

```
Enter value, char: 1.0543X          Enter value, char: 77.9555 G
Invalid character X                 Final value: 77955500000
```

```
void main() {
   double val;  // Input value
   char pref;   // Character representing prefix

   // Prompt for and read inputs
   printf("Enter value, char: ");
   scanf("%lf %c", &val, &pref);

   // Determine appropriate multiplier and change val accordingly

   switch (pref) {
   case 'k':
      val = val * 1000;
      break;

   case 'M':
      val = val * 1000000;
      break;

   case 'G':
      val = val * 1000000000;
      break;

   default:     // Error case
      printf("Invalid character %c\n", pref);
      return;
   }

   // Print final result after multiplication

   printf("Final value: %.0lf\n", val);
}
```