16.216: ECE Application Programming

Spring 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 if `x = 2` and `y = -2`?

```
switch (x + y) {
case -2:
  printf("neg ");
case 0:
  printf("zero ");
case 2:
  printf("pos ");
  break;
default:
  printf("def\n");
}
```

i. `neg`

ii. `zero`

iii. ***`zero pos`***

iv. `neg zero pos`

v. `zero pos def`

1 (continued)

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

```
i = 4;
  while (i <= 16) {
       printf("%d ", i);
       i = i * -2;
  }
```

i. Nothing

ii. 4

iii. 4 -8

iv. 4 -8 16

v. 4 -8 16 -32

c. Given the code sequence below:

```
int x, y;
do {
  scanf("%d %d", &x, &y);
} while ((x > 5) && (y < 5));
```

Which of the following possible input pairs will cause the do-while loop to end? In other words, which value(s) will cause the loop condition to be false?

A. 0 0
B. 4 1
C. 6 6
D. 10 5
E. 5 -2

i. A and B

ii. Only C

iii. D and E

iv. A, B, D, and E

v. All of the input pairs (A, B, C, D, and E)

1 (continued)

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

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

All 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() {
     int v1, v2;
     double d1, d2;

     d1 = 20.14;
     v1 = d1 + (7 + 1) / 3;        v1 = 20.14 + (7 + 1) / 3
                                      = 20.14 + 8 / 3
                                      = 20.14 + 2 = 22.14
                                      = 22 (truncated to int)

     d2 = v1 - 5 * 2;              d2 = 22 – 5 * 2
                                      = 22 – 10 = 12

     v2 = (v1 + 10) / (v1 - 10);   v2 = (22 + 10) / (22 – 10)
                                      = 32 / 12 = 2
                                        (truncated result from
                                        integer division)

     d1 = v2 + d1;                 d1 = 2 + 20.14 = 22.14

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

OUTPUT:

```
22
2 22.140000
12.000000
```

2 (continued)

b. (14 points)

```
void main() {
     int x = 9;
     double d1, d2, d3;

     d1 = (x / 2.0) + 3;        d1 = (9 / 2.0) + 3 = 4.5 + 3 = 7.5

     d2 = d1 / 10;              d2 = 7.5 / 10 = 0.75

     d3 = x - 3.5 + d2;         d3 = 9 – 3.5 + 0.75 = 5.5 + 0.75
                                   = 6.25

     x = d1 + d2 + d3;          x = 7.5 + 0.75 + 6.25
                                  = 14.5 = 14 (truncated to int)

     printf("%+07d\n", x);

     // NOTE: The last format specifier below uses a precision
     //   of 1 to print a double-precision value. In other
     //   words, you can read that specifier as
     //   "six point one L F."
     printf("%2.2lf %3.0lf %-6.1lf\n", d1, d2, d3);
}
```

OUTPUT: *Note that:*

- *For this problem, spaces are shown in red below.*
- *When printing d1, field width doesn't matter—specifying a precision of 2 will generate 4 output characters, and field width specifies a minimum number of characters to be printed.*
- *There are three spaces between 7.50 and 1. The first is simply the space between format specifiers; the next two are part of the field used to print the value 1, which is actually 0.75 rounded to the nearest integer.*
- *There are three spaces after the value 6.3, because that value is left justified within a field of six characters. The first three characters are the digits and decimal point; the next three characters are those spaces.*

```
+000014
7.50🟥🟥🟥1🟥6.3🟥🟥🟥
```

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. Each space between numbers is a single space character (`' '`). 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.

```
  16.2 2 19 -5
```

```
void main() {
     int v1, v2;
     double d1, d2;
     char ch1, ch2, ch3, ch4;

     scanf("%d%c %lf%c %d %c %lf %c",
          &v1, &ch1, &d1, &ch2,
          &v2, &ch3, &d2, &ch4);

     printf("%d %d\n", v1, v2);
     printf("%.2lf %.2lf\n", d1, d2);
     printf("%c%c%c%c\n", ch1, ch2, ch3, ch4);
}
```

This program reads its input as follows:

- *`v1` is the first integer value* → *`v1 = 16`*
- *`ch1` is the first character after 16* → *`ch1 = '.'`*
- *`d1` holds the next whole number* → *`d1 = 2`*
- *`ch2` is the first character after the 2 in "16.2"* → *`ch2 = ' '`*
- *`v2` holds the next integer* → *`v2 = 2`*
- *`ch3` is the first non-space character after 2* → *`ch3 = '1'`*
- *`d2` holds the next whole number* → *`d2 = 9`*
- *`ch4` is the first non-space character after `19`* → *`ch4 = '-'`*

<u>OUTPUT:</u> *Note that both double-precision values are printed with a precision of 2.*

```
16 2

2.00 9.00

. 1-
```

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. This program will prompt for and read a single character, then test that character to see if it is a letter, number, or neither. In each case, the program will reprint the character with an appropriate description, as shown in the test runs below (user input is underlined).

```
Enter character: E      Enter character: 6      Enter character: !
E is a letter           6 is a number           ! is neither
```

Hints: You can compare a character variable to a character constant. For example, the condition `(ch < '?')` is true for all characters with ASCII values higher than `'?'`. ASCII values are *usually* higher later in the alphabet—`'D'` is greater than `'C'`, but it's not greater than `'c'`.

Students were responsible for writing bold, underlined, italicized code.

```
void main() {
  char ch;

  // Prompt for/read input character
  printf("Enter character: ");
  scanf("%c", &ch);

  // Test for letter and print appropriate statement
  if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z'))
    printf("%c is a letter\n", ch);

  // Test for number and print appropriate statement
  else if (ch >= '0' && ch <= '9')
    printf("%c is a number\n", ch);

  // Handle all other cases
  else
    printf("%c is neither\n", ch);
}
```

3 (continued)

b. This program prompts the user to enter a single character command followed by three numbers: a pair of integers representing field width and precision, and a double-precision value to be printed. The command determines how the output value is formatted:

- If the command is 'F' or 'f,' use only the field width.
- If the command is 'P' or 'p,' use only the precision.
- For all other characters, use both field width and precision.

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

```
Enter char, 2 ints, and double: F 9 2 3.5
 3.500000                ← 3.5 is printed using field width of 9; precision of 2 is ignored

Enter char, 2 ints, and double: b 5 1 4.75
  4.8                    ← 4.75 is printed using field width of 5 and precision of 1
```

Students were responsible for writing bold, underlined, italicized code.

```
void main() {
  int fw, pr;       // Field width and precision
  char cmd;         // Single character command
  double val;       // Value to be printed

  // Prompt for/read input values
  printf("Enter char, 2 ints, and double: ");
  scanf("%c %d %d %lf", &cmd, &fw, &pr, &val);

  // Begin test of cmd
  switch (cmd) {

  // Use only field width
  case 'F': case 'f':
    printf("%*lf\n", fw, val);
    break;

  // Use only precision
  case 'P': case 'p':
    printf("%.*lf\n", pr, val);
    break;

  // Use both
  default:
    printf("%*.*lf\n", fw, pr, val);
  }
}
```

3 (continued)

c. This program should prompt for and read two integer input values. The program should then test the following three conditions, printing an appropriate message for each true condition:

- If v1 is greater than v2
- If v2 is negative
- If v2 divides evenly into v1

Note that more than one of these conditions may be true. If no conditions are true, the program generates no output other than the initial prompt. Your program output should match the sample program runs shown below (user input is underlined):

```
Enter inputs: 7 3         Enter inputs: -2 -2       Enter inputs: 9 3
7 is greater than 3       -2 is negative            9 is greater than 3
                          -2 divides into -2        3 divides into 9
```

```
void main() {
  int v1, v2;       // Input values

  // Prompt for/read two integers
  printf("Enter inputs: ");
  scanf("%d %d", &v1, &v2);

  // If v1 is larger value, print appropriate message
  if (v1 > v2)
    printf("%d is greater than %d\n", v1, v2);

  // If v2 is negative, print appropriate message
  if (v2 < 0)
    printf("%d is negative\n", v2);

  // If v2 divides evenly into v1, print appropriate message
  if ((v1 % v2) == 0)
    printf("%d divides evenly into %d\n", v2, v1);

}
```