16.216: ECE Application Programming

Spring 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 if **x** is 10?

```
switch (x) {
case 10:
   printf("A ");
case 20:
   printf("B ");
default:
   printf("C ");
}
```

Since there are no break statements in this switch statement, every printf() call will be executed if x is 10.

i. A

ii. B

iii. C

iv. A B

v. A B C

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

```
int i = 0;
while ((i * i) < 15) {
   printf("%d ", i);
   i = i + 2;
}
```

Loop runs while i squared is < 15, which means i must be less than 4. Values of i are actually printed, however.

i. `0`

ii. `0 2`

iii. `0 4`

iv. `0 2 4`

v. `0 4 16`

c. Which of the following if statements will print `"True"` if **x** is 216?

A.
```
if (x != 216)
   printf("True");
```
Condition is false

B.
```
if ((x < 220) && (x > 210))
   printf("True");
```
Condition is true since both parts are true

C.
```
if ((x < 200) || (x > 240))
   printf("True");
```
Condition is false since neither part is true

D.
```
if (x)
   printf("True");
```
Condition is true since value of x is non-zero

i. Only A

ii. Only B

iii. A and C

iv. B and D

v. All of the above (A, B, 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 didn't know we had lectures—I thought every day was a snow day."

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;
   int v2 = 35;
   double d1, d2;

   v1 = v2 / 10;                 v1 = 35 / 10 = 3 (int division)
   d1 = (v2 - 20.0) / 10;        d1 = (35 - 20.0) / 10
                                    = 15.0 / 10 = 1.5
   d2 = d1 + 2 * d1 - 7;         d2 = 1.5 + 2 * 1.5 - 7
                                    = 1.5 + 3 - 7 = -2.5
   v2 = d2 + d1;                 v2 = -2.5 + 1.5 = -1

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

OUTPUT (both d1 and d2 print with 6 digits after decimal point since precision is not specified):

3 -1 1.500000 -2.500000

2 (continued)
b. (14 points)

```
void main() {
   double d1, d2, d3;
   int x = 42;

   d1 = x / 8;                          d1 = 42 / 8 = 5.0 (int
                                          division truncates, then
                                          result assigned to double)

   d2 = (d1 + 8) / (d1 - 15);           d2 = (5.0 + 8) / (5.0 - 15)
                                           = 13.0 / -10.0 = -1.3

   d3 = d1 + d2 + 0.0975;               d3 = 5.0 + (-1.3) + 0.0975
                                           = 3.7975

   x = d3 - d2;                         x = 3.7975 - (-1.3)
                                          = 5.0975
                                          = 5 (truncates to int)

   printf("%d\n", x);
   printf("%.3lf\n", d1);               Print with precision 3

   printf("%.0lf\n", d2);               Print with precision 0
                                          (rounds -1.3 to -1)

   printf("%.2lf\n", d3);               Print with precision 2
                                          (rounds 3.7975 to 3.80)
}
```

OUTPUT:

```
5
5.000
-1
3.80
```

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. There are two spaces (`'  '`) between the `'6'` in `16.216` and the `'2'` in `2.23`, and two spaces between the `'3'` in `2.23` and the `'9'` in `950`.

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.

```
  16.216  2.23  950

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

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

   printf("%d %d\n", ival1, ival2);
   printf("%.3lf %.3lf\n", dval1, dval2);
   printf("%c%c%c\n", ch1, ch2, ch3);
}
```

This program reads its input as follows:

- *`ival1` is the first whole number* → *`ival1 = 16`*
- *`ch1` holds the first character after the `'6'` in `"16"`* → *`ch1 = '.'`*
- *`dval1` holds the next number* → *`dval1 = 216`*
- *`ch2` is the first character after the `'6'` in `"216"`* → *`ch2 = ' '`*
- *`dval2` holds the next number* → *`dval2 = 2.23`*
- *`ch3` is the first non-space character after `2.23`* → *`ch3 = '9'`*
- *`ival2` is the next whole number* → *`ival2 = 50`*

OUTPUT:

```
16 50
216.000 2.230
. 9
```

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 should first prompt for and read two pairs of numbers; each pair represents the lower and upper bounds of a range. The program should then do the following:

- If the ranges overlap, even if it's just at one endpoint each, print `"Overlap"`.
- If range 1 is lower (below low endpoint of range 2), print `"Range 1 < Range 2"`.
- If range 2 is lower, print `"Range 1 > Range 2"`.

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

```
Range 1: 1.5 3.5          Range 1: 3.14 3.24        Range 1: 16 216
Range 2: 2.7 8.7          Range 2: 3.25 9.25        Range 2: -2 15
Overlap                   Range 1 < Range 2         Range 1 > Range 2
```

Students were responsible for bold, underlined, italicized code.

```
int main() {
    double L1, H1;    // Endpoints of range 1
    double L2, H2;    // Endpoints of range 2

    // Prompt for and read ranges, value
    printf("Range 1: ");
    scanf("%lf %lf", &L1, &H1);
    printf("Range 2: ");
    scanf("%lf %lf", &L2, &H2);

    if (L1 <= H2 && L2 <= H1)
        printf("Ranges overlap\n");
    else if (H1 < L2)
        printf("Range 1 < Range 2\n");
    else
        printf("Range 1 > Range 2\n");
}
```

3 (continued)

b. Your code will print the grade represented by a score between 70 and 99, given these rules:

- Scores between 70-79 are in the C range, 80-89 in the B range, and 90-99 in the A range.
- If the second digit of the score is between 0-2 (for example, 70-72), add a '-' to the letter grade; if that digit is between 8-9 (for example, 88-89), add a '+' to the letter grade.

Rather than just checking possible grade ranges, your program should split the score into two digits and evaluate each separately. (Hint: 93 / 10 = 9, with a remainder of 3.)

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

```
Enter score: 83        Enter score: 79        Enter score: 91
Grade: B               Grade: C+              Grade: A-
```

Students were responsible for bold, underlined, italicized code.

```
void main() {
   int score;      // Score entered by user
   int hi, lo;     // High and low score digits

   // Prompt for and read score
   printf("Enter score:  ");
   scanf("%d", &score);

   // Separate out digits
   hi = score / 10;
   lo = score % 10;

   // Print first part of grade
   switch (hi) {
     case 9:
        printf("Grade: A");
        break;
     case 8:
        printf("Grade: B");
        break;
     case 7:
        printf("Grade: C");
        break;
   }

   // Print second part of grade
   if (lo >= 8)
     printf("+");
   else if (lo <= 2)
     printf("-");
}
```

3 (continued)

c. This program takes an integer between 1 and 23 representing a day in February thus far and prints an appropriate description. The notes below will help you identify the non-school days:
   - All Saturday dates are divisible by 7.
   - February 2, 3, 9, and 10 were all snow days.
   - February 16 was Presidents' Day.

Students were responsible for bold, underlined, italicized code.

```
void main() {
   int day;         // Day within February

   // Prompt for and read day
   printf("Enter day in February:  ");
   scanf("%d", &day);

   // Error: invalid date (not between 1 and 23)
   if (day < 1) || (day > 23)
      printf("Invalid date\n");

   // Weekend—check if day was Saturday or Sunday
   else if (day % 7 == 0) || (day % 7 == 1)
      printf("Weekend\n");

   // Other cases—write code around print statements that checks
   //   specific dates described above
   else {
      switch (day) {
         case 2: case 3:
         case 9: case 10:
            printf("Snow day\n");
            break;
         case 16:
            printf("President's Day\n");
            break;
         default:
            printf("School day\n");
      }
   }
```