16.216: ECE Application Programming

Spring 2013

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 choice you think best answers the question.

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

```
int x = 1;
while (x < 16) {
   x = x + 5;
   printf("%d ", x);
}
```

i. `6`

ii. `6 11`

iii. `1 6 11`

iv. `6 11 16`

v. `1 6 11 16`

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

```
int z = 1;
do {
   printf("z = %d\n", z);
   z = z + 1;
} while ((z % 3) != 0);
```

i. Nothing

ii. `z = 1`

iii. ***`z = 1`***
`z = 2`

iv. `z = 1`
`z = 2`
`z = 3`

c. Given the code sequence below:

```
int inval;
int n = 0;
do {
   scanf("%d", &inval);
   n = n + 1;
} while ((inval > 1) && (inval <= 7));
```

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

A. 0
B. 1
C. 5
D. 7
E. 9

i. A and B

ii. A and E

iii. ***A, B, and E***

iv. A, B, D, and E

v. 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 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) ***Expressions/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. (14 points)

```
void main() {
     int val1, val2;
     double doub1, doub2;

     doub1 = (5 + 8 / 3) / 2;        doub1 = (5 + 2) / 2 = 7 / 2
                                           = 3 (all int constants)
     val1 = doub1 - 2;               val1  = 3 – 2 = 1
     val2 = 4 * val1 + 1;            val2  = 4 * 1 + 1 = 4 + 1 = 5
     doub2 = 10.0 / (val2 - val1); doub2 = 10.0 / (5 – 1)
                                           = 10.0 / 4 = 2.5

     printf("%d\n%d\n", val1, val2);
     printf("\n%lf %lf\n", doub1, doub2);
}
```

Output: *Please note that the output does contain one completely blank line between the numbers 5 and 3.000000.*

```
1
5

3.000000 2.500000
```

2 (cont.)

b. (14 points)

```
void main() {
     double x = 16;
     double y = 4;
     double a, b, c;

     a = y / x;                 a = 4 / 16 = 0.25
     b = a * y + 0.138;         b = 0.25 * 4 + 0.138 = 1 + 0.138
                                  = 1.138
     c = b + 20;                c = 1.138 + 20 = 21.138

     printf("%06.2lf %-6.2lf\n", a, b);
     printf("%+10.4lf\n", c);
}
```

Output: *Note that:*

- *For this problem's output, spaces are shown in red below.*
- *There is a single space between 000.25 and 1.14*
- *1.14 is followed by 2 spaces, since this value is printed with a field width of 6, and the value contains 4 characters (3 digits and the decimal point).*
- *+21.1380 is preceded by 2 spaces, since this value is printed with a field width of 10, and the value contains 8 characters (6 digits, the decimal point, and the plus sign).*

```
000.25 1.14
  +21.1380
```

c. (12 points)

For this program, assume the user input is as follows, with one space between each number:

```
  12 34 56 78
```

```
void main() {
     char c1, c2, c3;
     int i1, i2, i3;

     scanf("%d%c", &i1, &c1);        i1 = 12, c1 = ' ' (since
                                       there is no space between
                                       the format specifiers)

     scanf("%d %c", &i2, &c2);       i2 = 34, c2 = '5' (because a
                                       space is between the format
                                       specifiers, spaces after
                                       the number 34 are skipped,
                                       and c2 holds the first non-
                                       space character after that)

     scanf(" %d %c", &i3, &c3);      i3 = 6, c3 = '7' (i3 holds
                                       the first number after '5';
                                       c3 holds the first non-
                                       space character after that)

     printf("%d %d %d\n", i1, i2, i3);
     printf("%c %c %c\n", c1, c2, c3);
}
```

Output: *Again, spaces are shown in red:*

- *There is a space between each pair of numbers on the first line.*
- *The output on the second line is as follows:*
  - *Two spaces (the value of c1, then the space between characters)*
  - *The number 5 (the value of c2), then the space between characters*
  - *The number 7 (the value of c3)*

```
12 34 6
  5 7
```

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 calculates a student's overall GPA after 4 semesters, given the GPA and credits per semester. The program prompts for and reads the four GPAs and credit counts, then calculates and prints the appropriate values, as in the example below (input is underlined):

```
Enter GPAs: 3.5 3.2 2.7 3.8
Enter credits: 14 17 18 15
Total credits: 64
Overall GPA: 3.27          ← NOTE: GPA printed using 2 decimal places
```

The overall GPA is based on a weighted average. For example, after 2 semesters, a student who earned a 3.5 GPA while taking 12 credits and a 3.0 GPA while taking 15 credits would have a GPA of (3.5 * 12 + 3.0 * 15) / (12 + 15) = 3.22.

Students were responsible for completing code that is underlined and in bold.

```
void main() {
   double G1, G2, G3, G4;      // Grade point averages
   int C1, C2, C3, C4;         // Credits per semester
   int total;                  // Overall total credits

   // Prompt for and read GPAs and credit counts
   printf("Enter GPAs: ");
   scanf("%lf %lf %lf %lf", &G1, &G2, &G3, &G4);
   printf("Enter credits: ");
   scanf("%d %d %d %d", &C1, &C2, &C3, &C4);

   // Calculate and print total credits and overall GPA
   total = C1 + C2 + C3 + C4;
   printf("Total credits: %d\n", total);
   printf("Overall GPA: %.2lf\n",
          (G1 * C1 + G2 * C2 + G3 * C3 + G4 * C4) / total);
}
```

3 (cont.)

b. 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 both L1 and H1 are inside the range specified by L2 and H2, print "`R1 in R2`".
- If both L2 and H2 are inside the range specified by L1 and H1, print "`R2 in R1`".
- If the previous conditions are false but the ranges do overlap, print "`Overlap`".
- If all previous conditions are false, print "`No overlap`".

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

```
Range 1: 4 8          Range 1: 1.2 5.6       Range 1: 0 5
Range 2: 5 7          Range 2: 3.4 7.8       Range 2: 6 10
R2 in R1              Overlap                No overlap
```

Students were responsible for completing code that is underlined and in bold.

```
void main() {
   double L1, H1;         // Endpoints of first range
   double L2, H2;         // Endpoints of second range

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

   // Test for range overlaps and print appropriate statements
   if ((L1 > L2) && (H1 < H2))
      printf("Range 2 contains Range 1\n");

   else if ((L2 > L1) && (H2 < H1))
      printf("Range 1 contains Range 2\n");

   else if ((H2 > L1) && (L2 < H1))
      printf("Ranges overlap\n");

   else
      printf("No overlap\n");
}
```

Note: *Because the problem specification wasn't specific enough, I accepted solutions in which interval endpoints are equal (for example, in the first if statement, checking:*

```
   if ((L1 >= L2) && (H1 <= H2))
```

3 (cont.)

c. This program should first prompt for and read two numbers representing a month and a year—for example, this month (February 2013) would be `2 2013`. The program then prints the number of days in the month. If the user enters an invalid month, print an error. Note that:

- February (month 2) has 28 days in most years, but 29 days in a leap year.
  - Leap years occur every 4 years, except when the year is divisible by 100 and not also by 400. For example, 2000 and 2012 were leap years; 1800 and 1900 were not.
- April (month 4), June (6), September (9), and November (11) have 30 days.
- All other months have 31 days.

Students were responsible for completing code that is underlined and in bold.

```
void main() {
   int month, year;  // Month and year

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

   // Check month and print appropriate number of days
   // Remember to check for leap year if month is February
   switch (month) {
   case 2:
      if ((year % 400 == 0) ||
          ((year % 100 != 0) && (year % 4 == 0)))
          printf("29 days\n");     // February in leap year
      else
          printf("28 days\n");     // February—no leap year
      break;

   case 4: case 6:  case 9: case 11:
      printf("30 days\n");          // April, June, Sept., Nov.
      break;

   case 1: case 3:
   case 5: case 7:
   case 8: case 10: case 12:
      printf("31 days\n");          // Jan., March, May, July,
                                    // August, October, December
      break;

   default:
      printf("Error: invalid month %d\n", month);  // Error case
   }
}
```