EECE.2160: ECE Application Programming

Spring 2018

Exam 1 Solution

1. (46 points) ***C input/output; operators***

a. (13 points) Show the output of the short program below 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.

```
int main () {
    int i;
    double d1 = 9.0;
    double d2, d3;

    i = 4.5 + d1 / 2;       i = 4.5 + 9.0 / 2 = 4.5 + 4.5 = 9.0 = 9
                                  (truncated to int)
    d2 = d1 / 100.0;        d2 = 9.0 / 100.0 = 0.09
    d3 = i + 0.21598;       d3 = 9 + 0.21598 = 9.21598
    d1 = d1 / 10;           d1 = 9.0 / 10 = 0.9

    printf("%d\n", i);
    printf("%.0lf\n%.3lf", d1, d2);
    printf(" %.4lf\n", d3);              // Note: 1 space before '%'

    return 0;
}
```

OUTPUT:

```
9
1
0.090 9.2160
```

1 (continued)

b. (13 points) For this program, assume the user inputs the line below. The digit `'2'` in `2+1.60` is the first character the user types. There is exactly one space (`' '`) between `2+1.60` and `20.18`.

You must determine how `scanf()` handles this input and then print the appropriate results, exactly as they would be shown on the screen. The program may not read all characters on the input line, but `scanf()` will read something into all seven variables declared in the program.

```
 2+1.60 20.18

int main () {
    int int1, int2;
    double d1, d2;
    char ch1, ch2, ch3, ch4;

    scanf("%lf %c%d %c %d%c%c %lf",
            &d1, &ch1, &int1, &ch2,
            &int2, &ch3, &ch4, &d2);

    printf("%d %d\n", int1, int2);
    printf("%.2lf %.2lf\n", d1, d2);
    printf("%c%c%c%c\n", ch1, ch2, ch3, ch4);

    return 0;
}
```

Solution:

d1 = 2
ch1 = '+' **(first non-space character after 2)**
int1 = 1
ch2 = '.' **(first non-space character after 1)**
int2 = 60
ch3 = ' ' **(first character after 60)**
ch4 = '2'
d2 = 0.18

OUTPUT:

```
1 60
2.00 0.18
+. 2
```

1 (continued)

a. (20 points) Complete this program, which 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 entering bold, underlined, italicized code.

```
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);
}
```

2. (34 points) ***Conditional statements***

a. (14 points) For the short program shown below, the first line of output (the prompt "`Enter val, 2 sets of endpoints: `") and the user input (`5.6 1.2 3.4 7.8 9.0`) is listed at the bottom of the page.

Complete the rest of the output for this program, given those input values.

```
int main() {
   double testval;
   double r1lo, r1hi;
   double r2lo, r2hi;

   printf("Enter val, 2 sets of endpoints: ");
   scanf("%lf %lf %lf %lf %lf",
          &testval, &r1lo, &r1hi, &r2lo, &r2hi);

   if (testval <= r1hi && r1lo <= testval)
      printf("R1\n");
   else {

      if (r2lo > testval)
         printf("Below R2\n");

      if (r2hi < testval)
         printf("Above R2\n");
      else
         printf("In R2?\n");
   }

   if (r1lo <= r2hi && r2lo <= r1hi)
      printf("Overlap\n");
   else
      printf("No overlap\n");
   return 0;
}
```

Condition is false, so else case executes

7.8 > 5.6, so "Below R2" prints

This if/else block is independent of previous one. 9.0 > 5.6, so else case executes, and "In R2?" prints

7.8 > 3.4, so second half of condition is false, overall condition is false, and "No overlap" prints

OUTPUT (the first line is given; write the remaining line(s)):

Enter val, 2 sets of endpoints: 5.6 1.2 3.4 7.8 9.0
Below R2
In R2?
No overlap

2 (continued)

b. (20 points) Complete this program, which implements two simple operations on a pair of integers. Your program should prompt for and read the operator and integers, then check for one of the conditions below, printing the appropriate output:

- If the operator is an uppercase or lowercase `'A'`, print the average of the integers.
- If the operator is an uppercase or lowercase `'M'`, print the higher (max) of the integers.
- In all other cases, print `"Error"`.

All numeric outputs should be printed with two digits after the decimal point. Three test cases are shown below, with user input underlined.

```
Enter input: A 5 10    Enter input: m 6 2.1   Enter input: x 3 5
Avg = 7.50             Max = 6.00             Error
```

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

```
void main()
   char op;               // Operator
   double v1, v2;         // Values to operate on
   double max;            // Maximum value for 'M'/'m' case

   // Prompt for and read expression
   printf("Enter input: ");
   scanf(" %c %lf %lf",&op, &v1, &v2);

   // Evaluate operator
   switch (op) {

      // Average
      case 'A': case 'a':

           printf("Avg = %.2lf\n", (v1 + v2) / 2);
           break;

      // Max value
      case 'M': case 'm':
           if (v1 > v2)   max = v1;
           else           max = v2;

           printf("Max = %.2lf\n", max);
           break;

      // Invalid op
      default:
           printf("Error\n");

   }
```

3. (20 points, 5 points each) ***While and do-while loops***

For questions 3a and 3b, circle or underline the one choice you think best answers the question.

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

```
int i = 4;
int j = 2;
while (i != j) {
   i = i + 4;          i = 8, then 12, then 16
   j = j * 2;          j = 4, then 8, then 16
   printf("%d %d ", i, j);
}
```

i. `4 2 8 4 12 8 16 16`

ii. `4 2 8 4 12 8`

iii. ***`8 4 12 8 16 16`***

iv. `8 4 12 8`

v. The loop prints nothing because the loop condition is initially false

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

```
int x = -3;
int y = 3;
do {
   printf("+ ");
   y = -x + 2;      y = 5,  then 4,  then 3, then 2
   x = x + 1;       x = -2, then -1, then 0, then 1
} while ((x < y) && (y > 2));
```

i. `+`

ii. `+ +`

iii. `+ + +`

iv. ***`+ + + +`***

v. `+ + + + +`

3 (continued)

c. Which pair of loops below produce the exact same output? **Circle TWO choices to answer this question.**

i.

```
int a = 1;
while ((12 % a) == 0) {
    printf("%d ", a);
    a = a * 2;
}
```

Loop while 12 is divisible by a
Prints: 1 2 4
Exits once a = 8

ii.

```
int b = 1;
do {
    printf("%d ", b);
    b = b * 2;
} while (b > 4);
```

Loop condition initially false,
but do-while guarantees 1 iter.
Prints: 1
Exits after 1 iteration (2 < 4)

iii.

```
int c = 4;
do {
    printf("%d ", 5 - c);
    c = c / 2;
} while (c > 0);
```

Prints: 1 3 4
c = 4 → 2 → 1 → 0

iv.

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

Prints: 1 2 4
Exits once d = 8

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 next exam is as easy as this question."

All of the above are "correct."

4. (10 points) ***EXTRA CREDIT***

REMEMBER, YOU CANNOT GET EXTRA CREDIT WITHOUT WRITING AT LEAST PARTIAL SOLUTIONS FOR ALL OTHER PROBLEMS ON THE EXAM.

Complete the program below, which reads an integer, `val`, copies its original value so it may be reprinted at the end, and then determines the number of digits in that integer. The variable `n` should hold the number of digits in `val` when the program is done. For example, if `val` = 5, `n` = 1; if `val` = 1033, `n` = 4.

The number of digits in `val` can be found by repeatedly dividing `val` until the result is 0; the number of steps required to reach that point is the number of digits in `val`. For example, if `val` = 16216, the program goes through the sequence below to determine `val` has 5 digits:

16216 → 1621 → 162 → 16 → 1 → 0 (each arrow represents one step)

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

```
int main() {
   int val, valCopy;      // Input value and its copy
   int n;                 // Number of digits in val

   printf("Enter number: ");   // Prompt for and read input value
   scanf("%d", &val);

   valCopy = val;
   n = 0;

   // COMPLETE PROGRAM AS DESCRIBED ABOVE
   do {
      n++;
      val = val / 10;
   } while (val != 0);

   // Print final results
   printf("Value: %d, # digits: %d\n", valCopy, n);
   return 0;
}
```