EECE.2160: ECE Application Programming

Spring 2019

Exam 1
February 22, 2019

Name: _____

Lecture time (circle 1): *8-8:50 (Sec. 201)* *12-12:50 (Sec. 202)*

For this exam, you may use only one 8.5" x 11" double-sided page of notes. All electronic devices (e.g., calculators, cell phones) are prohibited. Please turn off your cell phone ringer prior to the start of the exam to avoid distracting other students.

The exam contains 3 sections for a total of 100 points, plus a 10 point extra credit question. Please answer the questions in the spaces provided. If you need additional space, use the back of the page on which the question is written and clearly indicate that you have done so.

Please read each question carefully before you answer. In particular, note that:

- Questions 1c and 2b require you to complete short programs. We have provided comments to describe what each program should do and written some of the code.
    - Each program contains both lines that are partially written and blank spaces in which you must write additional code. **You must write all code required to make each program work as described—do not simply fill in the blank lines.**
    - Each test case is an example of how the program should behave in one specific case—**it does not cover all possible results of running that program.**
    - You can solve each of these questions using only the variables that have been declared, but you may declare and use other variables if you want.
- Carefully read the multiple choice problems. Questions 3a has exactly one correct answer, while Questions 3b and 3c may have more than one correct answer.
- **You cannot earn any extra credit without partial solutions to all parts of Questions 1, 2, and 3.** In other words, don't try the extra credit until you've tried to solve every other question on the exam.

You will have 50 minutes to complete this exam.

Q1: C input/output; operators	/ 46
Q2: Conditional statements	/ 34
Q3: While and do-while loops	/ 20
TOTAL SCORE	/ 100
Q4: EXTRA CREDIT	/ 10

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 leave 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, j, k;
   double d1, d2, d3;

   i = 3;
   d1 = 11.0;

   j = i * 1.5 + d1 / 2;
   k = j % (i + 1);

   d2 = d1 / 100 + 0.8;
   d3 = j + 0.32597;

   printf("%d %d %d", i, j, k);
   printf("\number: %lf\n", d1);
   printf("%.0lf %.3lf\n", d2, d3);

   return 0;
}
```

1 (continued)

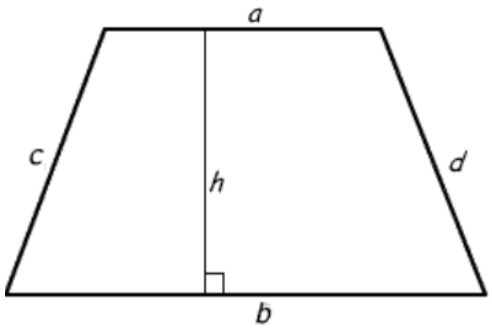
b. (13 points) For this program, assume the user inputs the line below. The digit `'1'` is the first character the user types. There is one space (`' '`) between the `'7'` in `1.97` and the `'$'`, one space between the `'8'` in `2.38` and the `'2'` in `20.6`, and two spaces between the `'6'` in `20.6` and the `'+'`. There is no space between `'+'` and `'5'`.

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 nine variables declared in the program. There are no formatting errors in the input!

```
     1.97 $2.38 20.6  +5

int main () {
   int int1, int2;
   double d1, d2, d3;
   char ch1, ch2, ch3, ch4;

   scanf("%d%c%d %c%lf%c%lf %c%lf",
          &int1, &ch1, &int2, &ch2, &d1,
          &ch3, &d2, &ch4, &d3);

   printf("%d %d\n", int1, int2);
   printf("%.2lf %.2lf %.2lf\n", d1, d2, d3);
   printf("%c%c%c%c\n", ch1, ch2, ch3, ch4);

   return 0;
}
```

1 (continued)

c. (20 points) Complete this program, which reads the dimensions of a trapezoid with base a, base b, side c, side d and height h, reprint the dimensions, and calculate and print the area and perimeter. Note:



- To calculate a trapezoid's area, add the two bases, multiply that sum by the height, and divide by two.
- To find the perimeter, add the bases and sides.

The formatting of your output should exactly match the formatting of the two test cases below, in which user input is underlined:

```
Enter a, b, c, d, h: 4 7 2.2 2.8 2
Base a: 4.00
Base b: 7.00
Side c: 2.20
Side d: 2.80
Height h: 2.00
Area: 11.000
Perimeter: 16.000
```

```
Enter a, b, c, d, h: 10 18 17.7 16.6 17
Base a: 10.00
Base b: 18.00
Side c: 17.70
Side d: 16.60
Height h: 17.00
Area: 238.000
Perimeter: 62.300
```

```
int main() {
   double a, b, c, d, h;    // Bases, sides, and height of trapezoid

   // Prompt for and read dimensions
   printf("Enter a, b, c, d, h: ");

   scanf("_____",_____);

   // Reprint dimensions

   printf("Base a: _____", _____);

   printf("Base b: _____", _____);

   printf("Side c: _____", _____);

   printf("Side d: _____", _____);

   printf("Height h: _____", _____);

   // Calculate and print area and perimeter

   printf("Area: _____", _____);

   printf("Perimeter: _____", _____);
   return 0;
}
```

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

a. (14 points) For the short program shown below, the first line of output (the prompt "`Enter 2 chars & int: `") and the user input (`Ex1`) is listed at the bottom of the page.

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

```
int main() {
   char c1, c2;
   int v1;
   printf("Enter 2 chars & int: ");
   scanf("%c%c%d", &c1, &c2, &v1);

   switch (c1) {
   case 'E': case 'e':
      printf("Exam\n");
   case 'T': case 't':
      printf("Test\n");
   default:
      printf("Neither\n");
   }

   switch (c2) {
   case '1':
      printf("%c%d\n", c2, v1);
      break;
   case 'x':
      printf("%d%c\n", v1, c2);
      break;
   default:
      printf("No match\n");
   }

   return 0;
}
```

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

Enter 2 chars & int: Ex1

2 (continued)

b. (20 points) Complete this program that reads three values and checks if they could represent the sides of a triangle, and, if so, if that triangle is equilateral, isosceles, or scalene. To test for each triangle type, your program should check the conditions below:

- If the sum of two sides is less than or equal to the third side, print “Triangle is not valid"
- Otherwise, if all sides are equal, print "Equilateral triangle "
- If only two sides are equal, print "Isosceles triangle "
- If no sides are equal, print "Scalene triangle"

Three test cases are shown below, with user input underlined.

```
Enter sides: 1 10 10      Enter sides: 2.2 2.2 2.2      Enter sides: 3.3 3 33.33
Isosceles triangle        Equilateral triangle          Triangle is not valid
```

```
int main() {
   double s1, s2, s3;          // Triangle side lengths

   /* Prompt for and read sides of a triangle */
   printf("Enter sides: ");

   scanf("_____________________",_____________________________________);

   // Check for possible results described above


      printf("Triangle is not valid");



      printf("Equilateral triangle");



      printf("Isosceles triangle");



      printf("Scalene triangle");

   return 0;
}
```

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

a. What is the output of the short code sequence below? **Choose only one answer.**

```
x = 9;
while (x > 1) {
   printf("%d ", x);
   x = x / 2;
}
```

i. No output—the loop condition is initially false

ii. `9`

iii. `9 4`

iv. `9 4 2`

v. `9 4 2 1`

b. 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 continue? In other words, which value(s) will cause the loop condition to be true? **This question has at least one correct answer, but may have more than one correct answer! Circle ALL choices that correctly answer the question.**

i. 0

ii. 1

iii. 5

iv. 7

v. 9

3 (continued)

c. Which loops below produce the following output?

```
* * * *
```

This question has at least one correct answer, but may have more than one correct answer! Circle ALL choices that correctly answer the question.

i.
```
int a = 4;
while (a < 0) {
   printf("* ");
   a = a - 1;
}
```

ii.
```
int b = 0;
do {
   b = b + 1;
   printf("* ");
} while (b <= 3);
```

iii.
```
int c = 7;
while (c) {
   c = c / 2;
   printf("* ");
}
```

iv.
```
int d = 1;
do {
   printf("* ");
   d = d * 3;
} while (30 > 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 hope the next exam is as easy as this question."

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 calculates a value that contains the digits of its input value, `inval`, in reverse. For example, if `inval = 456`, its reverse is `654`; if `inval = 2019`, its reverse is `9102`. The general algorithm requires you to isolate each digit, add it to a running total, then remove that digit from the remaining total. For example, the steps required for reversing 456 would be:

- Current digit = 6 → Running total = 6, remaining total = 45
- Current digit = 5 → Running total = 65, remaining total = 4
- Current digit = 4 → Running total = 654, remaining total = 0

```
int main() {
     int inval;          // Input value
     int dig;            // Current digit
     int res;            // Running total (becomes final result)

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

     // COMPLETE PROGRAM AS DESCRIBED ABOVE









     printf("Final result: %d\n", res);
     return 0;
}
```