EECE.2160: ECE Application Programming

Spring 2020

Exam 2

March 4, 2020

Name: ____

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

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 4 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 1 and 2b require you to complete a short program. We have provided comments to describe what your solution should do and written some code.
    - Each problem 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 function 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 4a and 4c may have more than one correct answer, while question 4b has exactly one correct answer.
- **You cannot earn extra credit without partial solutions to all parts of Sections 1, 2, 3, and 4.** In other words, don't try Section 5 until you've attempted every other question on the exam—"finishing the exam" doesn't have to include the extra credit problem!

You will have 50 minutes to complete this exam.

S1: Range checking	/ 25
S2: Switch statements	/ 40
S3: Loops	/ 15
S4: Multiple choice	/ 20
TOTAL SCORE	/ 100
S5: EXTRA CREDIT	/ 10

1. (25 points) ***Range checking***

Approximate values can be expressed with a margin of error—for example, 50 ± 5%, which implies a range from 47.5 to 52.5, since 2.5 is 5% of 50. This program should read an approximate value and error margin (as a percentage), as well as a value to be tested, and determines if the tested value is within the range of valid approximations. Two sample program runs are shown below, with user input underlined:

```
Enter approx. value, error margin, and test value: 50 5 49
49.00 is within +/- 5.00% of 50.00 ←OUTPUT USES 2 DECIMAL PLACES

Enter approx. value, error margin, and test value: 100 10 89.9
89.90 is not in range              ←OUTPUT USES 2 DECIMAL PLACES
```

```
void main() {
  double app;  // Approximate value (midpoint of valid range)
  double pct;  // Error margin expressed as a percentage
  double err;  // Error margin as an actual value
  double test; // Value to be tested

  // Prompt for/read approx. value, err. margin, and test value
  printf("Enter approx. value, error margin, and test value: ");

  scanf("______________________", ____________________________);

  // Calculate err (actual value of error margin)



  // Check if test is in range and print message if it is



    printf("________ is within +/- ____________ of ________\n",
      ______________________________________________________);

  // Handle case where value is not in valid range



    printf("_________ is not in range\n", _____________);
}
```

2. (40 points) ***Switch statements***

a. (15 points) For the short program shown below, the first line of output (the prompt "`Enter 2 integers & char: `") and the user input (`4 3 +`) is listed at the bottom of the page. Complete the rest of the output for this program, given those input values.

```
int main() {
    int i1, i2;
    char op;
    printf("Enter 2 integers & char: ");
    scanf("%d %d %c", &i1, &i2, &op);

    switch (op) {
    case '+':
        printf("i1 + i2 = %d\n", i1 + i2);
    case '-':
        printf("i1 - i2 = %d\n", i1 - i2);
    case '*':
        printf("i1 * i2 = %d\n", i1 * i2);
    case '/':
        if ( i2 == 0)
            printf(" the divisor cannot be 0!\n");
        else
            printf("i1 / i2 = %d\n", i1 / i2);
    default:
        printf("Wrong operator!\n");
    }

    switch (i1 % i2) {
    case 0: case 1:
        printf("ECE");
        break;
    case 2: case 3:
        printf("Programming");
        break;
    default:
        printf("ECE Application Programming");
    }
    return 0;
}
```

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

```
Enter 2 integers & char: 4 3 +
```

2. (continued)

b. (25 points) Complete this program, which prompts the user to enter a single character command followed by two integers. The command determines how the result is calculated:

- If the command is 'R' or 'r', the two integers represent the two sides of a rectangle, and the program calculates and prints the area of the rectangle.
- If the command is 'C' or 'c', the program treats the two integers as the radius and height of a cylinder and calculates and prints its area (3.14 * radius * height).
- For all other commands, the program prints the message "Invalid command".

Your solution must use a switch statement. Test cases are shown below:

```
   Enter char, 2 ints: R 2 3          Enter char, 2 ints: c 2 2
   area = 6.00                        area = 12.56
```

```
int main() {
   char op;
   int v1, v2;
   double area;

   // Prompt for and read inputs
   printf("Enter char, 2 ints: ");

   scanf("______________________", ____________________);

   // Evaluate operator

   switch (_______________________________________________) {

      // Rectangle


        printf("area = ________________\n", ________________);


      // Cylinder


        printf("area = ________________\n", ________________);


      // Invalid op


        printf("Invalid command\n");
    }
    return 0;
}
```

3. (15 points) ***Loops***
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 = 10;
   int j = 20;

   printf("Loop 1:\n");
   while (i < j) {
      printf("%d %d\n", i, j);
      i += 3;
      j -= 2;
   }

   printf("Loop 2:\n");
   for (i = 1; i <= 5; ++i)
      printf("%d\n", 5 - i);

   printf("Loop 3:\n");
   do {
      printf("%d\n", i);
      i = i * -2;
   } while (i < 5);

   return 0;
}
```

4. (20 points, 5 points each) ***Multiple choice***

a. 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 i;
for (i = 0; i < 12; i += 2) {
   printf("* ");
}
```

ii.

```
int a = 0;
do {
   a++;
   printf("* ");
} while (a < 5);
```

iii.

```
int c = 1;
while (c < 32) {
   c = c * 2;
   printf("* ");
}
```

iv.

```
int i;
for (i = 9; i > 0; i--) {
   if (i % 2)
     printf("* ");
}
```

4 (continued)

b. Assume you have two variables, `min` and `max`, representing the endpoints of a range. Which conditional statement below correctly tests if a third variable, `val`, is <u>outside the range defined by `min` and `max`</u>? **<u>This question has exactly one correct answer.</u>**

i.
```
if (val >= min && val <= max)
   printf("Out of range\n");
```

ii.
```
if (val >= min || val <= max)
   printf("Out of range\n");
```

iii.
```
if (val < min && val > max)
   printf("Out of range\n");
```

iv.
```
if (val < min || val > max)
   printf("Out of range\n");
```

4 (continued)

c. Your program contains two integer variables, `x` and `y`, for which the user must input values, as well as two more integers, `n` and `flag`, and a character variable named `junk`. You want to design an input validation loop that checks for two possible errors:

   (1) A formatting error (`scanf()` can't read the value of one or more inputs)
   (2) x is greater than y

Which of the input validation loops below correctly handles both error conditions and repeats until both inputs are error-free, with no issues that could cause the loop to run infinitely?

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

i.
```
do {
   printf("Enter x and y: ");
   n = scanf("%d %d", &x, &y);
   if (n < 2)
      printf("Format error\n");
   else if (x > y)
      printf("x > y\n");
} while (n < 2 || x > y);
```

ii.
```
do {
   printf("Enter x and y: ");
   n = scanf("%d %d", &x, &y);
   if (n < 2) {
      printf("Format error\n");
      do {
         scanf("%c", &junk);
      } while (junk != '\n');
   }
   else if (x > y)
      printf("x > y\n");
} while (n < 2 || x > y);
```

iii.
```
flag = 0;
do {
   printf("Enter x and y: ");
   n = scanf("%d %d", &x, &y);
   if (n < 2) {
      flag = 1;
      printf("Format error\n");
      do {
         scanf("%c", &junk);
      } while (junk != '\n');
   }
   else if (x > y) {
      flag = 1;
      printf("x > y\n");
   }
} while (flag == 1);
```

iv.
```
do {
   flag = 0;
   printf("Enter x and y: ");
   n = scanf("%d %d", &x, &y);
   if (n < 2) {
      flag++;
      printf("Format error\n");
      do {
         scanf("%c", &junk);
      } while (junk != '\n');
   }
   else if (x > y) {
      flag++;
      printf("x > y\n");
   }
} while (flag > 0);
```

4 (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. "I think the most recent programming assignments are still pretty easy."

   ii. "I think the programming assignments have gotten to be too difficult."

   iii. "I think the programming assignments have gotten harder, but are still fair."

   iv. "Is the semester over yet?"

5. (10 points) ***EXTRA CREDIT***

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

However, you can earn partial credit for a partial solution to this problem.

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;
}
```