EECE.2160: ECE Application Programming

Fall 2018

Exam 3

December 17, 2018

Name: ____

Lecture time (circle 1): *8-8:50 (Sec. 201)* *12-12:50 (Sec. 203)* *1-1:50 (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. If you have a cell phone, please turn off your ringer prior to the start of the exam to avoid distracting other students.

The exam contains 5 questions 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 1b, 1c, and 2a require you to complete short functions. We have provided comments to describe what each function should do and written some of the code.
  - Each function 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 function should behave in one specific case—**it does not cover all possible results of running that function.**
  - 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. Question 4b may have more than one correct answer, while questions 2b, 2c, and 3b each have exactly one correct answer.
- For this exam, unlike Exams 1 and 2, you may attempt the extra credit problem even if you have not at least partially completed all other problems on the exam.
- The extra pages accompanying the exam hold structure and function definitions, as well as hints for the extra credit question—info needed for questions 1a, 1b, 1c, 2a, and 5.

You will have 3 hours to complete this exam.

Q1: Structures	/ 32
Q2: File I/O	/ 28
Q3: Character & line input	/ 18
Q4: Bitwise operators	/ 22
TOTAL SCORE	/ 100
Q5: EXTRA CREDIT	/ 10

1. (32 points) ***Structures***

a. (12 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 I can easily recognize your final answer.

The `Elem` structure definition, as well as the `setElem()` function definition, are on the extra sheet provided with the exam.

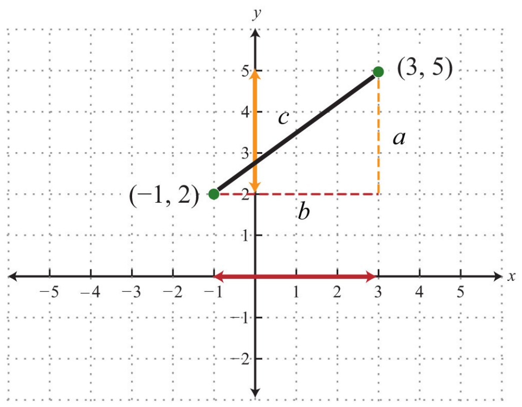
```
int main() {
     Elem arr[9];
     int i = 0;

     setElem(&arr[0], 'S', 4);
     setElem(&arr[2], 'l', arr[0].n + 1);
     setElem(&arr[3], 'n', 5);
     setElem(&arr[4], 'o', -2);
     setElem(&arr[6], 't', -5);
     setElem(&arr[7], 'u', -1);
     setElem(&arr[8], '\n', 1);

     arr[1] = arr[0];
     arr[1].c = 'S';
     arr[5] = arr[4];

     while (i < 9) {
          printf("%c", arr[i].c);
          i = i + arr[i].n;
     }

     return 0;
}
```

1 (continued)

b. (8 points) Complete the function below (the `Point` structure is defined on the extra sheet):

```
double dist(Point *p1, Point *p2);
```

This function takes pointers to two Point structures and finds the distance between them. The easiest way to find this distance is to treat the line connecting the points as the hypotenuse of a right triangle and use the Pythagorean theorem: given a right triangle with sides a and b and hypotenuse c, $a^2 + b^2 = c^2$.



For example, the picture to the right shows two points that could be represented by `Point` structures—say,

```
    Point v1 = {-1, 2};

    Point v2 = {3, 5};
```

If you call the function: `dist(&v1, &v2)`, the function will return 5, since $3^2 + 4^2 = 5^2$.

Given the relative simplicity of this function, you must declare your own variables (if necessary) when writing your solution below. You do not need to fill the available space—the function doesn't strictly require a lot of code. However, I'll accept any correct solution, regardless of how efficient it is.

Your solution can use the square root function from the `<math.h>` library, which simply returns the square root of its input argument as a double. For example, `sqrt(81)` returns 9.0.

```
double dist(Point *p1, Point *p2) {









}
```

The image shown above can be found at: https://2012books.lardbucket.org/books/beginning-algebra/s06-01-rectangular-coordinate-system.html

1 (continued)

c. (12 points) Complete the function below (the `Quadrilateral` structure is defined on the extra sheet): `int quadType(Quadrilateral *qp);`

This function takes one argument, a pointer to a `Quadrilateral` structure, and determines if the structure represents a rhombus (all 4 sides are of equal length), a parallelogram (all 4 sides are not equal, but opposite pairs of sides are equal), or neither of those two shapes.

The function returns 2 if the structure represents a rhombus, 1 if it represents a parallelogram, and 0 otherwise.

Your solution should use the `dist()` function defined in Question 1b. You do not need to write a correct solution to Question 1b to solve this problem—you simply need to call `dist()` properly in your solution to this function.

```
int quadType(Quadrilateral *qp) {
     double s1, s2, s3, s4;   // Side lengths for quadrilateral

     // Calculate the lengths of all 4 quadrilateral sides










     // If all 4 lengths match, the quadrilateral is a rhombus

     if (_______________________________________________________

         _______________________________________________________)
          return 2;

     // If all 4 lengths don't match, but opposite sides match,
     //    the quadrilateral is a parallelogram

     else if (__________________________________________________

              __________________________________________________)
          return 1;

     // Otherwise, it's neither
     else
          return 0;
}
```

2. (28 points) ***File I/O***
a. (20 points) Complete the function described below:

```
int maxFirst(char *infile, char *outfile);
```

This function works with two text files—one input, one output. The string arguments `infile` and `outfile` hold the names of these files.

The input file contains an even number of integers, which the function reads two at a time. After reading two values, the function prints those values to the output file, always printing the larger of the two values first. The function returns the total number of values read from the input file.

See the extra sheet for test cases that show sample file contents and function return values.

Assume the function always opens files successfully—do not check for any NULL pointers.

```
int maxFirst(char *infile, char *outfile) {
   FILE *infp, *outfp;    // File ptrs for input, output files
   int v1, v2;            // Input values
   int n = 0;             // Total # of inputs

   // Open input and output files

   infp = fopen(_______________________, ______________________);

   outfp = fopen(_______________________, _____________________);

   // Read input 2 values at a time until you reach end of file

   while (____________________________________________________) {

      // Print each pair of integers to output file with larger
      //   value printed first, then update count for # of inputs








   }
   // Close files





   // Return # of input values read

   return _______________;
}
```

2 (continued)

b. (4 points) You have a program that contains an array declared as:

```
double list[50];
```

Which of the following code snippets would correctly read a group of up to 50 values from a binary file and store them in this array? **<u>This question has exactly one correct answer.</u>**

i.
```
FILE *fp = fopen("input.bin", "rb");
fscanf(fp, "%lf", list);
```

ii.
```
FILE *fp = fopen("input.bin", "rb");
fread(list, sizeof(double), 1, fp);
```

iii.
```
FILE *fp = fopen("input.bin", "rb");
fread(list, sizeof(double), 50, fp);
```

iv. All of the above

c. (4 points) What is the appropriate way to test if your program has reached the end of a *<u>binary</u>* input file accessible through the variable `FILE *fpIn`? **<u>This question has exactly one correct answer.</u>**

i. Call `fread()` until the return value of that function is 0, since reaching the end of the file is the only reason `fread()` would return 0.

ii. Call `feof(fpIn)` before reading any data from the file.

iii. Call `feof(fpIn)` after attempting to read data from the file.

iv. Call `fscanf()` until that function returns EOF.

v. What's a binary file?

3. (18 points) ***Character and line input***

a. (14 points) Show the output of the short program below exactly as it will appear on the screen. The line below shows the prompt your program prints (**in bold**) and the input the user types (which is underlined):

Enter text: Today is December 17th.

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 I can easily recognize your final answer.

```
int e3_3a() {
   int n = 0;
   int i;
   char ch;
   char arr[50];
   printf("Enter text: ");
   while ((ch = getchar()) != 'D') {
      arr[n] = ch;
      ++n;
   }
   ungetc(ch, stdin);

   for (i = n - 1; i >= 0; i--)
      printf("%c", arr[i]);
   printf("\n");

   // Remember, isdigit(ch) returns true if ch is a digit
   while (!isdigit(ch = fgetc(stdin))) {
      printf("%c", ch);
   }
   printf("\n%c\n", ch);

   return 0;
}
```

3 (continued)

b. (4 points) Your program contains the following short piece of code:

```
char arr[10];
printf("Enter input line: ");
fgets(arr, 10, stdin);
printf("%s", arr);
```

The line below shows the prompt your program prints (**in bold**) and the user input (which is underlined). Assume the user presses Enter at the end of the line, thus putting a newline character at the end of the input:

```
Enter input line: The cow jumped over the moon.
```

What will the program print? **This question has exactly one correct answer.**

i. `The`

ii. `The cow`

iii. `The cow j`

iv. `The cow ju`

v. `The cow jumped over the moon.`

4. (22 points) ***Bitwise operators***

a. (14 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 I can easily recognize your final answer.

```
int main() {
   unsigned int q = 0xDEC17;
   unsigned int v1, v2, v3, v4;
   v1 = q | 0x1953C2;
   v2 = v1 ^ 0x0F0F0F0F;
   v3 = 0xDEADBEEF & ~v1;
   v4 = 0xEECE2160 | (v1 << 6);
   printf("%.6x\n", q);
   printf("%#.7x\n", v1);
   printf("%x\n", v2);
   printf("%#x\n", v3);
   printf("%x\n", v4);

   return 0;
}
```

4 (continued)

b. (4 points) Assume `unsigned int x = 0xABC123FF`.

Which of the following statements will set `x = 0xABC12300`? **<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. `x = x & 0xFFFFFF00;`

ii. `x = x | 0xFFFFFF00;`

iii. `x = x ^ 0x000000FF;`

iv. `x = ~x;`

v. `x = (x >> 8) << 8;`

c. (4 points) Circle one (or more) of the choices below that you feel best "answers" this "question," which has "nothing" to do with "bitwise operators."

i. "Thanks for the free points."

ii. "This is the best final exam I've taken today."

iii. "At least we're not here at 8:00 in the morning."

iv. "I forgot to circle the number of the section I'm enrolled in—I'll go back to the front page of the exam and do that now."

v. None of the above.

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

Everyone may attempt this problem, **even if you have not at least partially solved the rest of the exam.** Remember, you can earn partial credit for a partial solution to this problem.

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. *(Hint: The final output is easily readable, but you must show at least some work to get credit.)*

The contents of the three input files, as well as more hints, are on the extra handout.

```
int main() {
   FILE *fileA, *fileB, *fileC, *fp;
   char ch;
   char buf[50];
   int i, j, n;

   fileA = fopen("fileA.txt", "r");
   fileB = fopen("fileB.txt", "r");
   fileC = fopen("fileC.txt", "r");

   j = 0;
   while (fscanf(fileC, "%c%d", &ch, &n) != EOF) {
      if (ch == 'A')
         fp = fileA;
      else
         fp = fileB;

      for (i = 0; i < n; i++)
         buf[j++] = fgetc(fp);
   }
   buf[j] = '\0';

   for (i = 0; i < j; i++) {
      if (isdigit(buf[i]))
         buf[i] = 'a' + (buf[i] - 48);
      else if (tolower(buf[i]) == 'x')
         buf[i] = ' ';
   }

   printf("%s\n", buf);

   fclose(fileA);
   fclose(fileB);
   fclose(fileC);
   return 0;
}
```

5 (continued) ***<u>ADDITIONAL SPACE TO SOLVE EXTRA CREDIT PROBLEM</u>***