EECE.2160: ECE Application Programming

Fall 2018

Exam 3 Solution

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.

```
int main() {
     Elem arr[9];
     int i = 0;
     setElem(&arr[0], 'S', 4);                arr[0] = {'S', 4}
     setElem(&arr[2], 'l', arr[0].n + 1);     arr[2] = {'l', 5}
     setElem(&arr[3], 'n', 5);                arr[3] = {'n', 5}
     setElem(&arr[4], 'o', -2);               arr[4] = {'o', -2}
     setElem(&arr[6], 't', -5);               arr[6] = {'t', -5}
     setElem(&arr[7], 'u', -1);               arr[7] = {'u', -1}
     setElem(&arr[8], '\n', 1);               arr[8] = {'\n', 1}

     arr[1] = arr[0];                         arr[1] = {'S', 4}
     arr[1].c = 'S';      Here's the stupid typo I mentioned at
                            the exam—it should be arr[1].c = 'i'.
                            As is, this statement does nothing.
     arr[5] = arr[4];                         arr[5] = {'o', -2}

     while (i < 9) {
          printf("%c", arr[i].c);
          i = i + arr[i].n;
     }
     return 0;
}
```

Solution: The table below shows how the while loop generates output:

Output (arr[i].c)	Next i value (i + arr[i].n)		Output (arr[i].c)	Next i value (i + arr[i].n)
'S'	0 + 4 = 4		'S'	1 + 4 = 5
'o'	4 + -2 = 2		'o'	5 + -2 = 3
'l'	2 + 5 = 7		'n'	3 + 5 = 8
'u'	7 + -1 = 6		'\n'	8 + 1 = 9
't'	6 + -5 = 1			Loop ends

So, the final output is: `SolutSon` (yes, it should be `"Solution"`—stupid typo)

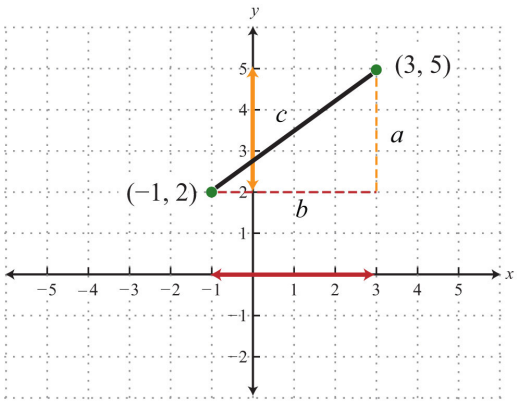
1 (continued)
b. (8 points) Complete the function below (the `Point` structure is defined on the extra sheet):

```
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) {
   double xdiff = p1->x - p2->x;
   double ydiff = p1->y - p2->y;
   return sqrt(xdiff * xdiff + ydiff * ydiff);
}
```

A couple of quick notes on the solution, based on questions I got during the exam:

- *What I'm calling "xdiff" is essentially "b" in the figure above, while "ydiff" is "a".*
- *A few people asked about using the absolute value function. However, since you're squaring the differences between the x-coordinates and y-coordinates, it really doesn't matter if that value is positive or negative.*
- *A couple of people also asked about using the power function (pow(x, y) returns x^y). It's generally more time-consuming to call that function than it is to simply multiply a number by itself if the exponent (2, in this case) is small.*

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
     s1 = dist(&qp->vert[0], &qp->vert[1]);
     s2 = dist(&qp->vert[1], &qp->vert[2]);
     s3 = dist(&qp->vert[2], &qp->vert[3]);
     s4 = dist(&qp->vert[3], &qp->vert[0]);

     // If all 4 lengths match, the quadrilateral is a rhombus
     if (s1 == s2 && s2 == s3 && s3 == s4)
          return 2;

     // If all 4 lengths don't match, but opposite sides match,
     //    the quadrilateral is a parallelogram
     else if (s1 == s3 && s2 == s4)
          return 1;

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

It doesn't matter which order you use the points, as long as you always use consecutive points, as described in the extra handout.

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(infile, "r");
   outfp = fopen(outfile, "w");

   // Read input 2 values at a time until you reach end of file
   while (fscanf(infp, "%d %d", &v1, &v2) != EOF) {

      // Print each pair of integers to output file with larger
      //   value printed first, then update count for # of inputs
      if (v1 > v2)
         fprintf(outfp, "%d %d\n", v1, v2);
      else
         fprintf(outfp, "%d %d\n", v2, v1);

      n += 2;
   }
   // Close files
   fclose(infp);
   fclose(outfp);

   // Return # of input values read
   return n;
}
```

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? **This question has exactly one correct answer.**

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 *binary* input file accessible through the variable `FILE *fpIn`? **This question has exactly one correct answer.**

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') {      Reads all characters
      arr[n] = ch;                         before 'D' in December,
      ++n;                                 storing each in arr[]
   }
   ungetc(ch, stdin);                     Places 'D' back in input

   for (i = n - 1; i >= 0; i--)           Prints characters from
      printf("%c", arr[i]);                arr[] in reverse order
   printf("\n");

   // Remember, isdigit(ch) returns true if ch is a digit
   while (!isdigit(ch = fgetc(stdin))) {  Reads up to 1st digit,
      printf("%c", ch);                    starting with 'D',
   }                                       and prints each char
                                           before first digit

   printf("\n%c\n", ch);                   Prints character that
                                           forced prior loop to
                                           end ('1')
   return 0;
}
```

OUTPUT:

```
 si yadoT                    There's a space before 's'
December                     There's a space after 'r'
1
```

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;
      = 0xDEC17 | 0x1953C2 =   0000 1101 1110 1100 0001 0111
                             OR  0001 1001 0101 0011 1100 0010
                                 0001 1101 1111 1111 1101 0111
                             = 0x1DFFD7

   v2 = v1 ^ 0x0F0F0F0F;
      = 0x1DFFD7 ^ 0x0F0F0F0F
      =    0000 0000 0001 1101 1111 1111 1101 0111
       XOR 0000 1111 0000 1111 0000 1111 0000 1111
           0000 1111 0001 0010 1111 0000 1101 1000
      = 0x0F12F0D8

   v3 = 0xDEADBEEF & ~v1;
      = 0xDEADBEEF & ~(0x1DFFD7) = 0xDEADBEEF & 0xFFE20028
      =    1101 1110 1010 1101 1011 1110 1110 1111
       AND 1111 1111 1110 0010 0000 0000 0010 1000
           1101 1110 1010 0000 0000 0000 0010 1000
      = 0xDEA00028

   v4 = 0xEECE2160 | (v1 << 6);
      = 0xEECE2160 | (0x1DFFD7 << 6)

   0x1DFFD7 << 6 = 0000 0000 0001 1101 1111 1111 1101 0111 << 6
                 = 0000 0111 0111 1111 1111 0101 1100 0000

   So, v4 =     1101 1101 1100 1101 0010 0001 0110 0000
             OR 0000 0111 0111 1111 1111 0101 1100 0000
                1101 1111 1111 1111 1111 0101 1110 0000
           = 0xEFFFF5E0

   printf("%.6x\n", q);          Print q with at least 6 digits
   printf("%#.7x\n", v1);        Print v1 with at least 7 digits
                                    and a leading 0x
   printf("%x\n", v2);           Print v2 with no leading 0x
   printf("%#x\n", v3);          Print v3 with a leading 0x
   printf("%x\n", v4);
   return 0;
}
```

OUTPUT for Question 4a

```
0dec17
0x01dffd7
f12f0d8
0xdea00028
effff5e0
```

4 (continued)

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

Which of the following statements will set `x = 0xABC12300`? **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. ***`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.

All of the above are "correct."

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;
}
```

Question 5 Solution: First, the while loop reads data from three text files:

- Each iteration, the loop reads one character and one integer from `fileC.txt`, storing those values in the variables `ch` and `n`, respectively. The loop repeats until the end of `fileC.txt` is reached.
- The body of the loop uses `ch` and `n` as follows:
  - `ch` determines if your program reads `fileA.txt` or `fileB.txt`—based on the letter it reads, it assigns the file pointer for that file to the variable `fp`.
  - `n` determines the number of characters read into the array `buf[]`.
- As noted in the extra handout, the file contents are as follows:

Name	Contents
FileA.txt	Yrx0w4rX2or
FileB.txt	ouns8sXr42t!
FileC.txt	A1B2A3B2A4B3A2B1A1B4

- So, here's how the program handles those files and fills `buf[]`:

Input from FileC.txt	FileA or FileB?	Chars read	Current state of buf[]
A1	FileA	Y	Y
B2	FileB	ou	You
A3	FileA	rx0	Yourx0
B2	FileB	ns	Yourx0ns
A4	FileA	w4rX	Yourx0nsw4rX
B3	FileB	8sX	Yourx0nsw4rX8sX
A2	FileA	2o	Yourx0nsw4rX8sX2o
B1	FileB	r	Yourx0nsw4rX8sX2or
A1	FileA	r	Yourx0nsw4rX8sX2orr
B4	FileB	42t!	Yourx0nsw4rX8sX2orr42t!

The for loop then goes through the array `buf[]` and modifies some of the characters:

- If `buf[i]` is a digit, it's changed to the actual integer it represents (`buf[i] - 48` does that, since 48 is the ASCII value of `'0'`) and then added to `'a'`.
  - The array contains one `'0'`, which changes to `'a'`, two `'2'`s, which change to `'c'`, two `'4'`s, which change to `'e'`, and one `'8'`, which changes to `'i'`.
- If `buf[i]` is `'X'` or `'x'`, it's changed to a space `' '`.

That loop therefore changes `buf[]` as shown below, with the string on the right representing the final output of the program:

`Yourx0nsw4rX8sX2orr42t!` → **Your answer is correct!**