ECS 30 Spring 2015

Homework 3 due (due: 6-5-2015) (June 5th, 2015) at 4:30pm

(late assignments will NOT be accepted)

Submit homeworks in the "ECS 30" homework box in 2131 Kemper Hall.

- do NOT submit them using "handin" from your CSIF accounts

-you can handwrite them (pencil or pen) or type them in a document; it just needs to be legible (if we can't read it, you will not get points for it)

At the top of your homeworks assignment, please include the following information:

Your Name
Your UC Davis ID number
ECS 30
Spring 2015

TA Tips:

1. These problems are from the "Review Questions" section, *not* "Programming Exercises".
2. If there are several lines of mathematical calculations, please underline or box the final solution.
3. If a command results in an error or seems invalid, write "Error" or "Invalid"
4. For explanation answers, usually 2-3 sentences are sufficient for a clear explanation. No more than 5 max.

Written assignment #2: Ch. 12: 4, 7; Ch. 13: 6, 11; Ch. 15: 1, 12; Ch. 16: 8, 14, Ch. 17: 1, 7, 13

The problems are reproduced below for your convenience:

Ch.12:

4. Describe the linear search and binary search algorithms in simple English.

7. Describe the steps that are involved in the selection sort algorithm.

Ch.13:

6. Assuming that variables of type *double* require eight bytes of memory and that a pointer requires four, draw a memory diagram showing a portion of the stack frame that contains the following declarations:

```
double d1;
double d2;
double *dp1;
double *dp2;
```

In your diagram, trace through the operation of these statements:

```
dp1 = &d1;
dp2 = &d2;
*dp1 = 3.14159;
d2 = 2.71828;
dp1 = dp2;
*dp1 -=d2;
```

11. What does the phrase *call by reference* mean?

Ch.15:

1. Which representation more closely corresponds to the internal representation of a text file: (a) a two-dimensional array consisting of a sequence of lines or)b_ a one-dimensional sequence of characters?

12. What is the purpose of the function *ungetc*?

Ch.16:

8. Suppose that the first two elements of *staff* have been initialized to contain the following values, just as in the chapter:

staff

0	Ebenezer Scrooge	Partner	271-82-8183	250.00	1
1	Bob Cratchit	Clerk	314-15-9265	15.00	7

Starting with the variable *staff*, how would you select the field corresponding to Bob Cratchit's salary? How would you select Ebenezer Scrooge's first initial?

14. When you want to store a database in a file, you must define both an internal and external representation. Why are both representations necessary? What factors must you consider in the design of each one?

TA comment: Write a list of Internal factors and a list of External Factors for part 2 of this question.

Ch.17:

1. Define the term *recursion.*

7. What are the fundamental operations of a queue?

13. What is the computational complexity of the binary-search algorithm as implemented in Figure 12-3?

FIGURE 12-3 FindStringInSortedArray

```
/*
* Function: FindStringInSortedArray
* Usage: index = FindStringInSortedArray(key, array, n);
* ------------------------------------------------------
* This function returns the index of an element that matches key
* in the specified array of strings, which must be sorted in
* lexicographic order.  If key appears more than once in the
* array, the function can return any index at which it appears.
* If key does not appear at all in the first n elements
* of the array, FindStringInSortedArray returns -1.
*
* This implementation uses the "binary search" algorithm.  At
* each stage, the function computes the midpoint of the remaining
* range and compares the element at that index position to the
* key.  If there is a match, the function returns the index.
* If the key is less than the string at that index position, the
* function searches in the first half of the array; if the key is
* larger, the function searches in the second half of the array.
*/

int FindStringInSortedArray(string key,
                            string array[],
                            int n)
{
   int lh, rh, mid, cmp;

   lh = 0;
   rh = n - 1;
   while (lh <= rh) {
      mid = (lh + rh) / 2;
      cmp = StringCompare(key, array[mid]);
      if (cmp == 0) return (mid);
      if (cmp < 0) {
         rh = mid - 1;
      } else {
         lh = mid + 1;
      }
   }
   return (-1);
}
```

Thus, to search an array of N elements requires N comparisons if you