<u>COP 3502 Suggested Program Edits: DMA, Linked Lists (Week 2 Programs)</u>

1) To either arrayallocation.c or arrayallocation2.c, add a function that takes in the dictionary of words, and a word, and prints out all the words in the dictionary that are anagrams of the input word. (An anagram is a rearrangement of the letters in a word. So "CARE" is an anagram of "RACE".)

2) To either arrayallocation.c or arrayallocation2.c, add a function that takes in the dictionary of words, and a string, which represents the letters available to use to make a sign. This string may have repeats. The function should return the number of words in the dictionary that can be formed with the letters in the input string. For example, if the second string is "THERART", then words such as "THREAT", "ART" and "RATHER" should be counted, but "HAH" should not. (Note that there is only one 'H' in "THERART" but "HAH" has 2 H's. For the other three words, there are enough copies of each letter in the string "THERART" to form the word.) Note that the solution to this problem should be a relatively small edit to the solution to problem #1.

3) Add basic error checking to gridgame.c, which uses static memory allocation. Also, instead of adding an X to each location, make sure an X only prints in the current location.

4) To the file dynarrayofpointers.c, add a function that takes in pointers to two struct points and returns the Manhattan distance between the two points. Write another function which takes in a pointer to a struct point and a double pointer to a struct point (storing an array of points) and returns the longest Manhattan distance between the first point and any of the points in the array. Test your function with a small set of points.

5) Add a function to dynarrayofstruct.c which takes in an array of struct, its length, a boolean value (set to 0 to represent x, and set to 1 to represent y), and an integer and returns how many points in the array have either an x or y coordinate equal to the last integer. Here is the function prototype:

```
// Returns the number of points in array that have the appropriate coordinate
// equal to target. If isY == 1, then the # of points where the y coordinate
// is equal to target is return. If isY == 0, then the # of points where the
// x coordinate is equal to target is returned.
int numPtsOnLine(struct point array[], int size, int isY, int target);
```

Test your function appropriately.

6) To the file bigintadd.c add a compare function, which takes in two big integers, returns a negative integer if the first one is smaller than the second, returns 0 if they are equal, and returns a positive integer if the first one is bigger than the second one.