2) (10 pts) DSN (Linked Lists)

Consider using a linked list to store a string, where each node, in order, stores one letter in the string. The struct used for a single node is included below. Write a function that takes in two pointers to linked lists storing 2 strings, and prints out the letters in the string in alternating order, starting with the first letter of the first string. If one string runs out of letters, just skip over it. For example, if the two strings passed to the function were “hello” and “computer”, then the function should print “hceolmlpouter”.

```
typedef struct node {
    char letter;
    struct node* next;
} node;

void printMixed(node* word1, node* word2) {

    while (word1 != NULL || word2 != NULL) {          // 2 pts

        if (word1 != NULL) {                          // 1 pt
            printf("%c", word1->letter);              // 2 pts
            word1 = word1->next;                      // 1 pt
        }

        if (word2 != NULL) {                          // 1 pt
            printf("%c", word2->letter);              // 2 pts
            word2 = word2->next;                      // 1 pt
        }
    }

}
```

Grading: If a student follows the method above, it should be fairly easy to award points. There are many other ways to solve this however, most of which will probably take more code. Here is an alternate "idea scheme" for grading responses that use a different approach:

1 pt – attempting to iterate through both lists.
1 pt – advancing a pointer through word1
1 pt – advancing a pointer through word2
2 pts – some sort of artifact to get alternating behavior
2 pts – printing each letter in each list (1 pt for each list)
3 pts – avoiding NULL pointer issues

It is likely that partial credit might be awarded for that last criteria of avoiding NULL pointer issues (ie in some places the solution avoids them, but not all). Give partial on this as you see fit.