1) (10 pts) DSN (Recursive Coding)

Ten tiles each have strings of in between 1 and 4 letters on them (hardcoded in the code below). The goal of this problem is to complete the code below so it counts the number of different orders in which **all** of the tiles can be placed such that the string they form creates a palindrome (a word that reads the same forwards and backwards). All of main, as well as the function which determines if a particular ordering of the tiles forms a palindrome, **included on the next page** have been given. You may call this function in the function go. Complete the recursive function (named go) to complete the solution.

```
#include <stdio.h>
#include <string.h>
#define N 10
#define MAXLEN 5

int go(int perm[], int used[], int k, char tiles[N][MAXLEN]);
int eval(int perm[], char tiles[N][MAXLEN]);
char MYTILES[N][MAXLEN] = {"at","ta","g","cc","ccac","ca","cc","gag","cga","gc"};

int main(void) {
    int perm[N];
    int used[N];
    for (int i=0; i<N; i++) used[i] = 0;
    int res = go(perm, used, 0, MYTILES);
    printf("Number of tile orderings that create palindromes is %d\n", res);
    return 0;
}

int go(int perm[], int used[], int k, char tiles[N][MAXLEN]) {

    if (k == N)
        return _____________________________________________ ;

    int res = 0;

    for (int i=0; i<N; i++) {

        if (used[i]) continue;

        used[i] = ____ ;

        perm[k] = ____ ;

        ____________________________________________________ ;


        used[i] = ____ ;
    }

    return res;
}
```

```
int eval(int perm[], char tiles[N][MAXLEN]) {

    char tmp[N*MAXLEN];
    int idx = 0;
    for (int i=0; i<N; i++) {
        int len = strlen(tiles[perm[i]]);
        for (int j=0; j<len; j++)
            tmp[idx++] = tiles[perm[i]][j];
    }
    tmp[idx] = '\0';

    for (int i=0; i<idx/2; i++)
        if (tmp[i] != tmp[idx-1-i])
            return 0;

    return 1;
}
```