1) (10 pts) DSN (Recursive Coding)

Finish the function below so that it determines the **number of permutations** of size ***n*** (values 0 through ***n***-1, inclusive) such that each pair of adjacent values is within ***maxgap*** of each other. You can assume any spot in the permutation that has not been filled in yet will be 0. The function should also take in an array, ***perm***, the permutation, a second array, ***used***, which indicates the values that have been used in the permutation, and an integer, ***k***, representing which is the current empty spot (0-indexed) to be filled in. **Functions that fail to utilize recursion will receive 0 points**. (For example, if ***n*** = 4, ***maxgap*** = 2 and ***k*** = 0, the only permutations of size 4 that would not be counted are the ones that have 1 and 4 adjacent, since the difference between these is 3, which is bigger than maxgap.)

```
int kClosePerm(int* perm, int* used, int n, int maxgap, int k) {


    if (n == k)

        return 1; ;                    // 1 pt

    int res = 0;
    for (int i=0; i<n; i++) {

        if (used[i]) continue;                              // 1 pt

        if (k>0 && abs(perm[k-1]-i) > maxgap) continue;     // 2 pts

        used[i] = 1;                                        // 1 pt
        perm[k] = i;                                        // 1 pt
        res += kClosePerm(perm, used, n, maxgap, k+1);      // 3 pts
        used[i] = 0;                                        // 1 pt

    }

    return res;
}
```