Summer 2021 **Algorithms and Analysis Tools Exam, Part B**

Name: ____________________, UCFID: ____________________, NID: ________________

1) (10 pts) DSN (Recursive Coding)

We call a list of positive integers **nice** if each pair of consecutive elements shares a common divisor greater than 1. For example, the list 18, 15, 35, 40 is nice because 18 and 15 are both divisible by 3, 15 and 35 are both divisible by 5, and 35 and 40 are both divisible by 5 as well. Given a list of unique positive integers, complete the recursive code below so that it counts the number of permutations of an original list that are nice lists. (For example, for the 4 numbers above, there are 12 nice arrangements: [18,15,35,40], [18,15,40,35], [18,40,15,35], [18,40,35,15], [15,18,40,35], [15,35,40,18], [35,15,18,40], [35,15,40,18], [35,40,18,15], [35,40,15,18], [40,18,15,35], and [40,35,15,18].)

The strategy the code below uses is to go through each permutation of the integers 0, 1, 2, …, n-1, where n is the number of values being considered. For each permutation, the evaluation function returns 1 if the order of the values creates a nice list, and 0 otherwise. Over all permutations, these values are added. Note: the gcd function provided returns the greatest common divisor shared by the 2 positive integer parameters.

```
int numArr(int values[], int perm[], int used[], int k, int n) {

    if (k == n)

        return eval(values, perm, n);

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

        if ( !used[i] ) {

            used[i] = 1;

            perm[k] = i;
            res += numArr(values, perm, used, k+1 , n );

            used[i] = 0;
        }
    }
    return res;
}

int eval(int values[], int perm[], int n) {

    for (int i=0; i< n-1 ; i++)

        if ( gcd( values[perm[i]] , values[perm[i+1]] ) == 1 )
            return 0;
    return 1;
}

int gcd(int a, int b) {
    return b == 0 ? a : gcd(b, a%b);
}
```

Grading: 1 pt per slot, gcd slots can be exchanged. Slot must be perfect to get point.