COP 3502 Section 2 Final Exam - Part D (Recursion, Sorting, Bitwise Ops/Backtracking)

Date: 7/29/2020
Start Time: 5:30 pm EST
End Time: 6:00 pm EST

Directions: Please type up answers in either a Word Document (.doc, docx) or a Text Document (.txt) or scan your written work to a .pdf file and upload your document AND SUBMIT IT to the appropriate assignment in Webcourses COP 3502 Section 2. Don't recopy the questions due to the time constraints, but clearly label which question number your work corresponds to. On the document you submit, put your first and last name in the top left hand corner. On the following line, write "My Exam 1 Part D Answers", centered. Following that, place your answers, numbered, in order (1, 2, 3).

1) (5 pts) Consider the problem of finding the mode (most frequently occurring value) in an array. John attempts to solve the problem recursively. His strategy is to recursively call his mode function on the left half of the array and the right half of the array, and then use these answers to calculate the mode of the whole array. Assuming that his function only returns the mode and nothing else, why is his strategy doomed to fail?

2) Sorting

(a) (5 pts) In a Merge Sort of 8 elements, the Merge function gets called 7 times. Consider a Merge Sort being executed on the array shown below. What does the array look like right **AFTER** the sixth call to the Merge function completes?

index	0	1	2	3	4	5	6	7
value	40	27	12	18	11	99	31	16

Index	0	1	2	3	4	5	6	7
Value								

(b) (5 pts) Consider sorting the array below using a Bubble Sort. Show the contents of the array after each iteration of the algorithm completes. (Note: after the first iteration, the maximum array value should be in its correct spot.)

Index	0	1	2	3	4	5
Original	33	18	22	9	15	14
After 1st						
After 2nd						
After 3rd						
After 4th						
After 5th						

3) (10 pts) For the purposes of this question, a permutation of size n is any ordering of the integers 0, 1, 2, …, n-1. We define a spaced-out permutation of size n to be a permutation such that two consecutive terms in the permutation differ by at least 2. For example, [0, 2, 4, 1, 3] is a spaced out permutation of size 5, and [5, 2, 4, 0, 3, 1] is a spaced out permutation of size 6, but [3, 0, 2, 1] is not a spaced out permutation since the 2 and 1 are adjacent and differ by only 1. Fill in the blanks below so that all spaced out permutations of size N are printed out. (Note: You may use the abs function from the math library. The function takes in a single integer and returns its absolute value.)

```
#include <stdio.h>
#include <math.h>
#define N 6

int main(void) {
    int perm[N], used[N];
    for (int i=0; i<N; i++) used[i] = 0;
    printSpaced(perm, 0, used);
    return 0;
}

void printSpaced(int perm[], int k, int used[]) {

    if (k == N) {
        for (int i=0; i<N; i++) printf("%d ", perm[i]);
        printf("\n");
        return;
    }

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

        if ( ____ || ________________________________________ )) {

            used[i] =  ____;

            perm[k] =  ____;

            printSpaced( ______, ______, ______);

            used[i] = 0;
        }
    }
}
```