COP 3502 Section 1 Quiz #1 Version B (SLMP, Dynamic Memory Allocation) Solution

1) (5 pts) Rewrite the following statement using the function calloc instead:

```
int* nums = malloc(sizeof(int)*n);
```

```
int* nums = calloc(n, sizeof(int));
```

Grading: 1 pt LHS, 1 pt calloc, 1 pt n as first param, 1 pt sizeof(int) as second param, 1 pt for having exactly 2 params

2) (10 pts) Complete the function below so that it returns 1 if there exist two different numbers in the array list with a difference of target, and 0 otherwise.. In order to earn full credit, the function must run in O(n) time.

```
// Pre-condition: list is length n, sorted, with unique values.
// Post-condition: Returns 1 if two different values in list have a
//                 difference of target, and 0 otherwise.
int sub_to_target(int list[], int n, int target) {

    int i = 0, j = 0;

    while ( j<n ) {                                  // 1 pt

        if (list[j]-list[i]<target)                  // 2 pts
            j++;                                     // 1 pt

        else if (list[j]-list[i]>target)             // 2 pts
            i++;                                     // 1 pt

        else                                         // 1 pt
            return 1;                                // 1 pt
     }

     return 0;                                       // 1 pt
}
```

Grading note: order of the if clauses can be any order. Last clause is worth 2 total instead of 3…

Max grade for $O(n^2)$ is 6 out of 10. (1 pt setup, 1 pt outer loop, 2 pts inner loop (1 pt only if allows repeat val), 2 pts check inside and return 1, 1 pt return 0 outside

3) (12 pts) Write a function that takes in a single integer n, and dynamically allocates an array of n strings, each of length n+1. The function should then fill the 2D character array with a checkerboard pattern with the character 'X' in the top left corner (row 0, column 0), and the character 'O' in all spaces adjacent (up, down, left, right) to any character 'X'. In addition, the function should put the null character ('\0') at the end of each string. Finally, the function should return a pointer to the character array created. For n = 3, the array of strings should store {"XOX", "OXO", "XOX"}.

```
char** checkerboard(int n) {

    char** res = calloc(n, sizeof(char*));        // 2 pts
    for (int i=0; i<n; i++) {                     // 1 pt
        res[i] = calloc(n+1, sizeof(char));       // 2 pts
        for (int j=0; j<n; j++) {                 // 1 pt
            if ( (i+j)%2 == 0)                    // 1 pt
                res[i][j] = 'X';                  // 1 pt
            else                                  // 1 pt
                res[i][j] = 'O';                  // 1 pt
        }
        res[i][n] = '\0';                         // 1 pt
    }
    return res;                                   // 1 pt
}
```

4) (8 pts) The code below shows a struct definition and then a dynamically allocated array. Assuming that that is followed by code that dynamically allocates the necessary memory inside each struct, write the corresponding code to free that memory.

```
typedef struct q4 {
    int* array;
    char x;
} q4;

int n;
scanf("%d", &n);
q4* ptr = calloc(n, sizeof(q4));
// Allocate memory for ptr[0]…ptr[n-1].

// *** ANSWER QUESTION FREE ALL DYNAMICALLY ALLOCATED MEMORY ***

for (int i=0; i<n; i++)          // 2 pts
    free(ptr[i].array);          // 5 pts - 1 pt free, 1 pt ptr[i]
                                 // 2 pts dot, 1 pt array
free(ptr);                       // 1 pt
```