COP 3502 Section 4 Quiz #1 Version D (SLMP, Dynamic Memory Allocation) Solution

1) (5 pts) Allocate an array of integer pointers so that it contains 567890 pointers, and set each of those pointers to NULL.

```
int** array = calloc(567890, sizeof(int*)); // OR

int** array = malloc(567890*sizeof(int*));
for (int i=0; i<567890; i++) array[i] = NULL;
```

Grading: calloc solution – 1 pt LHS, 1 pt calloc, 1 pt 56789, 1 pt sizeof, 1 pt 2 params
malloc solution – 3 pts first line 2 pts last line

2) (10 pts) Complete the function below so that it returns 1 if there exist two different numbers in the array list such that the larger number divided by the smaller number equals target, and 0 otherwise. You may assume that list has all distinct positive values less than 30000 and that target is also less than 30000. (Note that if integers a*b = c, then when you divide c by b you get a exactly.) In order to earn full credit, the function must run in O(n) time.

```
// Pre-condition: list is length n, sorted, with unique positive ints
//                less than 30000. 1 < target < 30000.
// Post-condition: Returns 1 if two different values in list are such
//                 that when the larger is divided by the smaller it
//                 equals target exactly, or 0 otherwise.
int div_to_target(int list[], int n, int target) {

    int i = 0, j = 0;

    int i = 0, j = 0;

    while ( j<n ) {                                  // 1 pt

        if (list[i]*target > list[j])                // 2 pts
            j++;                                     // 1 pt
        else if (list[i]*target < list[j])           // 2 pts
            i++;                                     // 1 pt
        else                                         // 1 pt
            return 1;                                // 1 pt
     }

     return 0;                                       // 1 pt
}
```

3) (12 pts) The Vigenere square is a 26 by 26 square where the first row has all 26 letters in order, 'A' through 'Z'. The second row has the 26 letters in order starting with 'B', wrapping around to 'A' at the end, etc. (The start of the first 5 rows is "ABCDE", "BCDEF", "CDEFG", "DEFGH" and "EFGHI", respectively. The last row starts "ZABCD".) Write a function that dynamically allocates an array of 26 strings, each size 27, copies the contents of the square into the array, including the null character ('\0') in the last position of each row, and returns a pointer to the memory allocated. (For full credit a double for loop structure must be used and the only character literal that can appear in the code is 'A'. 0/12 will be given if all 26 rows are explicitly written out.")

```
char** vigeneresquare() {


    char** box = calloc(26, sizeof(char*));            // 3 pts
    for (int i=0; i<26; i++) {                         // 1 pt
        box[i] = calloc(27, sizeof(char));             // 2 pts
        for (int j=0; j<26; j++)                       // 1 pt
            box[i][j] = (i+j)%26 + 'A';                // 3 pts
        box[i][26] = '\0';                             // 1 pt
    }

    return box;                                        // 1 pt

}
```

Grading: Note if calloc is used then seting the null character isn't necessary, so in that case the calloc for box[i] is worth 3 pts.

4) (8 pts) Write code that calls the vigeneresquare function, storing what it returns in a pointer called `sqptr` then prints out each row of the square (there are 26 and you can hard code that number), and then frees all the associated memory for the square.

```
char** sqptr = vigeneresquare();           // 2 pts

for (int i=0; i<26; i++)                   // 1 pt total for both
    printf("%s\n", sqptr[i]);              // 2 pts

for (int i=0; i<26; i++)
    free(sqptr[i]);                        // 2 pts
free(sqptr);                               // 1 pt
```