1) (10 pts) ALG (Algorithm Analysis)

Consider the following function:

```
int* makeArray(int n) {
    int* array = calloc(n, sizeof(int));
    int i, j;
    for (i=0; i<n; i++)
        for (j=i; j<n; j = j+i+1)
            array[j]++;
    return array;
}
```

(a) (1 pt) Assuming that the function is called with a value of n = 12 or greater, what will array[11] store when the array is returned from the function?

6 (**Grading:** 1 pt all or nothing)

(b) (3 pts) In general, what will array[k] store when the function completes, assuming the function was called with an input value of k+1 or greater?

array[k] will store the number of divisors of k+1.
(**Grading**: 3 pts mostly all or nothing, perhaps partial credit if there is some mention of divisibility but if the answer isn't correct.)

(c) (2 pts) Write a summation that provides a tight upper bound on the number of times the line of code array[j]++ runs when the function is called with the input value n.

$$\sum_{i=1}^{n} \frac{n}{i}$$

(**Grading:** 1 pt bounds, 1 pt function inside sum)

(d) (4 pts) Utilizing the fact that $\sum_{i=1}^{n} \frac{1}{i} = O(lgn)$, determine the run time of the function makeArray for an input of size n. (Note: This run time is equal to the summation from part c.)

$$\sum_{i=1}^{n} \frac{n}{i} = n \sum_{i=1}^{n} \frac{1}{i} = nO(lgn) = O(nlgn)$$

(**Grading:** 2 pts pulling out n, 1 pt plugging in given info, 1 pt final answer)