Honors Computer Science I (COP 3502) Exam #1
Date: 1/25/2024

Name: _____

1) (a) (6 pts) Determine the following sum in terms of n:

$$\sum_{i=n+1}^{n^2} 2i$$

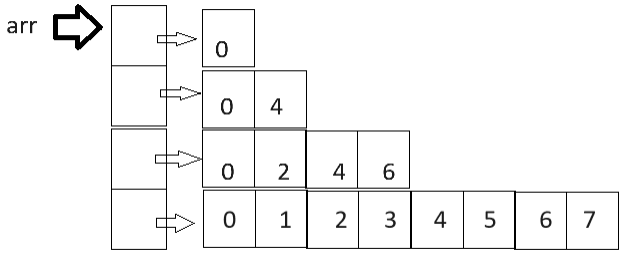
(b) (4 pts) Write a short function that runs in $O(n^2)$ that takes in n as a parameter and returns the result of the summation above by actually adding each of the terms individually. The function prototype is below. (Your grade is based on the direct translation of the summation, not returning the correct answer.)

```
int f(int n) {




}
```

2) (15 pts) Find an exact closed form solution for the following recurrence relation. Your answer should be of the form $T(n) = \frac{(an+b)3^n+c}{d}$, where a, b, c and d are all integers.

$$T(n) = T(n-1) + n3^{n-1}$$
$$T(0) = 0$$

4) (18 pts) A power array of size n is a 2D array that contains n+1 arrays, where array i is of size 2^i. Each of these n+1 arrays stores values equally spaced out, starting at 0. The common difference of each array is half of the previous array with the last array having a common difference of 1. Here is a picture of the power array of size n = 3



Write a function that takes in a positive integer n, allocates the correct amount of memory for the array, fills it, and returns a pointer to the array of arrays. Do NOT use the pow function. Use only integers in your calculation, multiplying ints in a loop as needed. (Hint: you should keep track of two counters. One counter initializes at 1, the other at 2^n. After each outer loop iteration, the first counter will double and the second will divide by 2. These two numbers store all you need to fill up a row of the table.

```
int** powerarray(int n) {




}
```

5) (6 pts) A program that solves a traveling salesman problem on n locations implements an algorithm with a run-time of $O(n2^n)$. For n = 16, the program takes 75 ms to complete. How long, **in seconds,** would we expect the algorithm to take when run on an input with size n = 20?

_____

6) (6 pts) What is the run-time of the function below in terms of the input variables r and c? Please give your answer in Big-Oh format. Please justify your answer for full credit.

```
int f(int** a, int r, int c) {

    int s = 0;
    for (int i=0; i<r; i++) {
        int x = 0, y = c-1;
        while (x < y) {
            int t = rand()%2;
            int m = (x+y)/2 ;
            if (t == 0)
                x = m+1 ;
            else
                y = m-1 ;
        }
        s += a[i][x] ;
    }
    return s ;
}
```

_____

7) (17 pts) The union of two sets is the set of all items that belong in either of the two sets. One way to store a set is in an array in sorted order. Write a function that runs in O(n+m) time that takes in one array of size n and another array of size m and creates a new array of the appropriate size that stores the union of the two input sets and returns a pointer to the newly created array. For example, if the input arrays were [2, 3, 6, 12] and [1, 2, 6, 8, 12, 14] the array created and returned would be [1, 2, 3, 6, 8, 12, 14]. (Hint: At first you won't know the exact size, but you can allocate "plenty of room", then before returning, resize the array to the right size.)

```
int* setunion(int* set1, int n, int* set2, int m) {

}
```

8) (3 pts) The C programming language was built off of its predecessor, which just happens to be the same as the predecessor to the letter C in the English alphabet. What was the name of the language that the C programming language was built off of?

_____

C Language Reference

```
// Allocates size bytes of uninitialized storage.
void* malloc(size_t size);

// Allocates a block of memory for an array of num elements,
// each of them size bytes long, and initializes all of its
// bits to 0.
void* calloc(size_t num, size_t size);

// Reallocates the given area of memory. It must be previously
// allocated by malloc, calloc or realloc and not yet freed
// with a call to free or realloc.
// This is done by either expanding or contracting the existing
// area pointed to by ptr, if possible. If not, a new memory
// block of size new_size bytes is allocated, copying memory
// area with size equal the lesser of the new and the old sizes,
// and freeing the old block. If there is not enough memory,
// the old memory block is not freed and null is returned.
void* realloc(void* ptr, size_t new_size);
```

Summation Formulas

$\sum_{i=1}^{n} c = cn \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$, , $\sum_{i=0}^{\infty} x^i = \frac{1}{1-x}, |x| < 1$, $\sum_{i=0}^{n-1} x^i = \frac{x^n-1}{x-1}, x \neq 1$

Scratch Work: Please clearly mark any work below you would like graded.