This problem set is worth a total of 100 points (plus 10 optional bonus points). Each ★ represents 5 points and each ♦ represents 10 points. Partial credit will be given when appropriate, provided that you show your work and not only the final solution. This assignment is individual and you must turn in your own write-up. Please clearly describe your answers in the order they are given. Solutions should be turned in using your account on Gradescope in PDF form.

1. ♦★ Solve each of the following recurrences using the iteration method:
   
   (a) \( T(1) = 1, \quad T(n) = 2T(n/2) + n \)
   
   (b) \( T(1) = 4, \quad T(n) = T(n - 1) + 4 \)
   
   (c) \( T(0) = 1, \quad T(n) = T(n - 1) + 2^n \)
   
   (d) \( T(0) = 1, \quad T(n) = 2T(n - 1) \)
   
   (e) \( T(1) = 1, \quad T(n) = T(n/3) + 1 \)
   
   (f) \( T(1) = 1, \quad T(n) = T(n - 1) + n - 1 \)

2. ♦ Consider the following C++ code:

   ```cpp
   int foo(int n) {
       int result = 0;
       if (n == 0) {
           return result;
       }
       for (int i = 0 ; i < n ; i++) {
           result += i;
       }
       return foo(n/2) + result + foo(n/2);
   }
   ```

   Find and solve the recurrence for the time complexity of `foo`.

3. ♦ Given the n-queens algorithm shown in-class. Draw the call tree (graphic representation of all the calls), for an initial call of this algorithm with \( n = 4 \) queens.

4. ♦ Write a C++ recursive function `long int tribonacci(int n)` that returns the \( n \)-th number in the Tribonacci sequence, where each number in the sequence is the sum of the previous three numbers. Consider the following base cases: \( T_1 = 1, T_2 = 1, T_3 = 1 \).

5. ♦ Write a C++ recursive function `void reverse(char *str, int n)` that will reverse a sequence of characters of length \( n \). Your function should complete with at most \( \lfloor n/2 \rfloor + 1 \) recursive calls.
6. ◊ Write a C++ recursive function `bool palindrome(char *str, int n)` that will return `true` if 
`str` of length `n` is a palindrome and `false` otherwise. Your function should complete with at most 
`\lceil n/2 \rceil + 1` recursive calls.

7. ◊ Write a C++ recursive function `bool is_sorted(int *array, int n)` that determines if the 
sequence of `n` integers in `array` is sorted in ascending order.

8. ◊ Given the following declaration:

   ```cpp
   int n, *p1, *p2;
   double *p3;
   ```

Mark the following statements as `True` (correct) or `False` (incorrect):

   (a) `p1 = p2;`
   (b) `n = p1;`
   (c) `p3 = p1;`
   (d) `*p3 = *p2;`
   (e) `*p1 = *p2;`
   (f) `n = *p2;`
   (g) `p1 = &p2;`
   (h) `p1 = &n;`
   (i) `n = &p1;`

9. ◊ What is the output of the following C++ code:

   ```cpp
   int *p, *q;
p = new int [10];
q = p + 10;
for (int i = 0 ; i < 10 ; i ++) {
    *(p+i) = i;
}
while (q != p) {
    q --;
    *q *= 2;
}
for (int i = 0 ; i < 10 ; i ++) {
    std::cout << p[i] << " ";
}
   ```

10. ◊ Given an array `A`, of `n` integers, describe a method to find the longest subarray of `A` such that all 
the numbers in that subarray are in sorted order. What is the running time of your algorithm?