1. List four kinds of resources that might be limited, and that we might therefore wish to measure when software is being developed or used. (One of these is “running time.”)
    - Running time
    - Memory used by data (storage space)
    - Memory used by code
    - Time to code

    Which of these will we be focussing on in this course?
    - Running time
    - Memory used by data (storage space)
2. Describe two ways that we might use to measure ”running time.”
    - Run the code and time the execution
    - Analyze the code
3. Briefly describe each of the following. What are the strengths, and weaknesses of each of the first two? Under what circumstances might the third be of interest?
    a. Worst Case Analysis
        i. Advantages:
            - Upper bound on running time (guarantee that the algorithm will not take any longer for *any* inputs of the given size)
            - For some algorithms, worst-case occurs fairly often (eg. searching an array for an element not in it)
        ii. Disadvantage:
            - For some cases, the worst case rarely occurs (eg. array in reverse order is the worst case for one variation of quicksort)

    b. Average Case Analysis
        i. Advantage:
            - Captures resource consumption for typical inputs
        ii. Disadvantages:
            - Executions on some inputs of the given size can take *much* longer than the average case
            - May be difficult to determine what the average case actually is - some assumption about the distribution of the inputs is *always* needed

    c. Best Case Analysis

        Occasionally of interest, but usually together with other measures(eg. see whether best and worst cases running times are close).
4. Recall that we introduced the notion of a “loop variant” when we studied proofs of correctness of algorithms, and saw that these were useful when we wanted to prove that algorithms halt.

    Describe how they are useful for the worst-case analysis of the running time of an algorithm as well.

    The number of executions of loop body is *less than or equal to* the value of f_L (loop variant function) immediately before execution of the loop begins.

5. What is a recurrence? How is it used when one is trying to bound the worst-case running time of an algorithm?

   A recurrence is when a program calls itself a constant number of times. In a *recurrence relation*, T(n) is expressed using the same function T evaluated as **smaller** inputs. Explicit (non-recursive) values of T are given for small inputs *n* (base cases).

6. Briefly describe how to find an upper bound for the number of steps used by a program, in the worst case, if it is executed when its precondition is satisfied, in each of the following cases.
   a. The program consists of a single assignment statement or is a continue statement.

      1 unit per operator (eg. single arithmetic/Boolean operation, comparison, or assignment)

   b. The program is an if-then-else statement.

      Worst-case running time is T (worst-case running time to evaluate the if condition) + the maximum worst-case running time between the two branch paths.

   c. The program is a while loop.

      If the worst-case cost to evaluate the loop condition G is $\leq T_1$, and the worst-case cost to execute the loop body S is $\leq T_2$, then assuming the loop is executed at most *k* times, the total worst-case running time for the loop is $(k+1)T_1 + kT_2$.

   d. The program is a sequence of two smaller programs.

      Assuming a sequence comprised of S_1; S_2, if the worst-case running time of S_1 is T_1, and the worst-case running time of S_2 is T_2, then the total worst-case running time of the entire program is at most $T_1 + T_2$.

   e. The program is one of the other tests or control structures provided by Java — specifically, an if-then statement, or a do-while or a for loop.
      i. For an if-then statement, you simply add the worst-case running time of the condition + the body of the conditional (assume conditional executes).
      ii. For a do-while loop or a for loop, it is possible to re-write them as while loops, and then use the method for calculating the worst-case running time of a while loop.

   f. The program includes a main method that calls other methods (included in the program), but recursion is not used.

      Consider it as a sequence of programs.

   g. The program is a “simple” recursive method — it only calls itself, (at most) some fixed number of times.

      The upper bound is written using a *recurrence relation*, which expresses T(n) using the same function T evaluated at **smaller** inputs.

7. How can you find a lower bound for the number of steps used by a program, in the worst case, if it is executed when its precondition is satisfied?

   In order to prove that the worst-case running time of a program *P* is *at least T*, for input size *N* (for a fixed *N*):

   - Find a valid input *I* of size *N* (where "valid means that *P*'s precondition is satisfied).
   - Counter the number of steps used by *P* on input *I*
   - *If* this number is greater than or equal to *T then* you have proved what we want.