CS210 Data Structures and Algorithms
Second Concept Assignment (20 marks plus 2 bonus marks)
Due date: October 25 at 11:55 pm
Important: No late concept assignments will be accepted

Please submit on OWL a pdf file or an image file with your solution to the assignment. You must also submit the completed java class Odd.java. You are encouraged to type your answers. If you decide to submit handwritten answers to the questions please make sure that the TA will be able to read your solutions. If the TA cannot read your answers you will not be given credit for them.

Remember that concept assignments must be submitted by the due date; **no late concept assignments will be accepted unless you have an academic accommodation**. You might find this fact useful:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$

1. (1 mark) Consider a hash table of size $N = 7$ where we are going to store integer key values. The hash function is $h(k) = k \bmod 7$. Draw the table that results after inserting, in the given order, the following values: 3, 10, 8, 43, 0. Assume that collisions are handled by separate chaining.

2. (1 mark) Show the result of the previous exercise, assuming collisions are handled by linear probing.

3. (4 marks) Repeat exercise (1) assuming collisions are handled by double hashing, using a secondary hash function $h'(k) = 5 - (k \bmod 5)$.

4. (4 marks) Consider the following algorithm.

   Algorithm proc(A, n)
   In: Array A of even size n
   if n = 0 **then return**
   else {
   proc$(A, n-2)$
   for $i \leftarrow 0$ **to** $n-1$ **do** $A[i] \leftarrow A[i] + i$
   }

   The time complexity of this algorithm is given by the following recurrence equation, where n is even:

   $f(0) = c_0$
   $f(n) = f(n-2) + c_1 n + c_2$, for $n > 0$

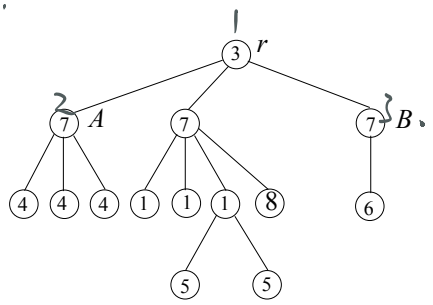
   where c_0, c_1, and c_2 are constants. Solve the recurrence equation **for n even and** give the order of $f(n)$. You **must** explain how you solved the equation.

5. (6 marks) Complete the provided Java class Odd.java by designing and implement in Java an algorithm `numOdd(r)` that receives as input the root r of a tree and it outputs the number of internal nodes of the tree that have odd degree. To test whether a number `k` is odd, you can use this statement: `if ((k % 2) == 1)` then `k` is odd.

   For example, for the tree below with root r the algorithm must output the value 3, as nodes r, A, and B have odd degree. For the tree with root r' the algorithm must return the value 0.

   If you want to test your program you can use the provided classes Node.java and TestOdd.java; this latter class will invoke your algorithm on the tree with root r shown in the figure. We will perform additional tests on your algorithm.

cont



5.(ii) (3 marks) Compute the worst case time complexity of your algorithm as a function of the total number n of nodes in the tree. You must explain how you computed the time complexity and give the order of the time complexity of the algorithm

6. (3 marks) Compute the time complexity of the following algorithm. You must explain how you computed the time complexity and you must give the order of the time complexity.

Algorithm algo(A, B, n)
In: Arrays A and B of size $\frac{n(n+1)}{2}$
  $i \leftarrow 0$
  $j \leftarrow 0$
  while $j < \frac{n(n+1)}{2}$ **do** {
    $B[i] \leftarrow A[j]$
    $i \leftarrow i + 1$
    $j \leftarrow j + i$
  }