CS210a Data Structures and Algorithms
Second Concept Assignment (20 marks)
Due date: October 26 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 Degree.java. You are encouraged to type your answers. If you decide to submit hand-written 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.

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 values. The hash function is $h(k) = k \bmod 7$. Draw the table that results after inserting, in the given order, the following values: 12, 5, 3, 24, 35. 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 $h'(k) = 5 - (k \bmod 5)$.

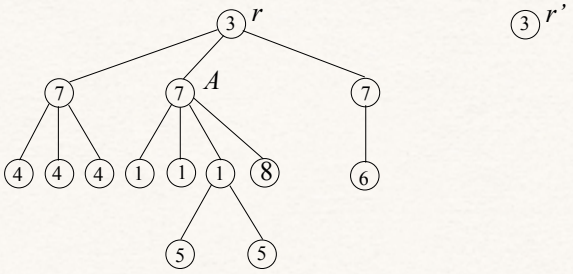
4. (3.5 marks) Solve the following recurrence equation using repeated substitution **and** give the order of $f(n)$. You must show how you solved the equation.

   $f(0) = 7$

   $f(n) = f(n-1) + 3n$

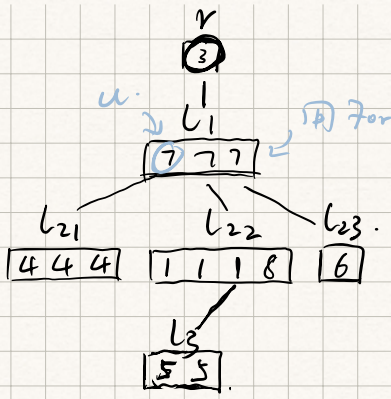
5.(i) (7 marks) Complete the provided Java class Degree.java by designing and implement in Java an algorithm an algorithm `maxDegree(r)` that receives as input the root r of a tree and it outputs the maximum degree, or maximum number of children of any node in the tree. For example, for the tree below with root r the algorithm must output the value 4, as node A has 4 children and no node has more than 4 children; for the tree with root r' the algorithm must return the value 0.

   You can download from OWL classes Node.java and TestNode.java that you can use to test your algorithm.



5.(ii) Compute the worst case time complexity of your algorithm as a function of the total number n of nodes in the tree. You must

- (1 mark) explain what the worst case for the algorithm is
- (2 marks) explain how you computed the time complexity
- (0.5 marks) give the order of the time complexity of the algorithm



for each 遍历每一个 Node u.

getchildren return child.

numChildren return num.

is Leaf return T/F.

getValue return v.

```
function (Node[] a) {
    遍历 a.   count=0.
all a[i] = null → return 0
    a[x] = Node[].  Count +1.
      function (a[x]).
    return Count + function (a[x]).
```