Problem Set 1

1. Prove that $\log n! = \theta(n \log n)$. Use this to find out if $\lceil \log n \rceil!$ and $\lceil \log \log n \rceil!$ are polynomially bounded. A function $f(n)$ is said to be polynomially bounded if there exists an integer k such that $f(n) = O(n^k)$.

2. Iterated logarithmic function is defined as $\log^*(n) = min\{i \geq 0 : \log^i n \leq 1\}$. For example, $\log^* 16 = 3$ where log is taken w.r.t base 2. Which is aymptotically larger: $\log(\log^* n)$ or $\log^*(\log n)$?

3. Write an efficient algorithm that checks whether a given singly linked list contains a loop. A loop is a sequence of nodes $v_1, v_2, \ldots, v_k$ such that $v_1 \to v_2 \to \ldots \to v_k \to v_1$.

4. Consider a stack with an additional operation, $MULTIPOP(S, k)$ which removes the k top objects of stack S, popping the entire stack if the stack contains fewer than k objects. The cost of the opearation $MULTI-POP(S, k)$ is k, while that of $PUSH(S, x)$ and $POP(S)$ is 1. Now consider a sequence of n stack operations on an initially empty stack, where each operation is either $PUSH, POP$ or $MULTIPOP$. Prove that the total cost of these n operations is $\theta(n)$.

5. Design a data structure $SpecialStack$ that supports all the stack operations like $push(), pop(), isEmpty(), top()$ and an additional operation $getMin()$ which should return minimum element from the $SpecialStack$. All these operations of $SpecialStack$ must be $O(1)$. To design $SpecialStack$, you should only use standard Stack data structure and no other data structure like arrays, list, etc.

6. Describe a $O(n)$-time algorithm that, given a set S of n integers in sorted order and another integer x, determines whether or not there exist two elements in S whose sum is exactly x.

7. Given an unsorted array A of size n that may contain duplicates and a number $k < n$, design an $O(n)$ algorithm that returns $true$ if array contains duplicates within a distance of k, ie. there exists $i, j \in [0, n-1]$ such that $A[i] = A[j]$ and $|i - j| < k$.