1. Give definitions to: graph, connected graph, connected component. Describe breadth first search algorithm. Describe depth first search. Minimum spanning tree. MST algorithms for unweighted graph.
2. Implement a doubly linked list (Java, pseudocode). Describe following functions:

* Remove(T value);
* GetNth(int i);
* hasLoop(); - which detects if loop in the list (propose the solution)

1. Algorithm complexity. Ways to estimate complexity. Big-O notation: O(g(n)), Θ(g(n)), Ω(g(n)). Simplification rules. Well-known sorting algorithm complexities (insertion sort, quicksort, count sort, mergesort, heapsort). Ways to measure time in a program.
2. Implement a data structure (Java, pseudocode) for a **directed** graph. Implement functions for adjacent nodes search.Implement an **efficient** function friendToEveryone(node):boolean that acts as a witness that **every** node is adjacent to some node. Which is the best way to organize the graph in this case?
3. Definition of recursion. Recursion implementation in operating system. Call stack. Tail recursion. Pitfalls of recursion. O(n) recursive implementation for Fibonacci numbers.
4. Describe different implementations of a Stack. What are their comparative advantages or disadvantages in terms of speed? Provide linked stack memory overhead estimation.
5. Sorting. Insertion sort. Selection sort. Write selection sort and theoretically estimate it’s complexity. Counting sort. Counting sort complexity.
6. Provide a data structure with O(1) for the operations of: insert, remove, contains, get random element. Show why these operations have O(1) complexity.
7. Map abstract data type. Hash table. Hash function. Avalanche property. Collision. Collision solving techniques. Probing.
8. Estimate time complexity of **foo()** method

public static int **bar**(int k) { B(k)

int result = 0;

while (--k > 0) result \*= k;

return result;

}

public static int **foo**(int n) { F(n)

if (n == 1)

return 1;

return bar(n - 1) / foo(n - 1);

}