Practice Questions for theory exam: For code questions, practice on paper. Test your code on PC only when you are confident it will work without errors. For some questions, you can reuse ADTs and code you may have already written (e.g. LinkedList ADT, BinarySearchTree ADT, Stack, Queue).

Q: Suppose you are given pointer P to a node in a doubly linked list. Write code to insert a new node N such after insertion, N is the immediate previous node to P.

Q: Write code to traverse a linked list and print its elements. Assume that the linked list has a header node (which can be skipped for printing).

Q: Suppose B is a binary search tree. Let x be the key value at root of the tree. Assume you are given a sequence of keys S = {x,a_1,a_2 … a_k}. This sequence represents keys of the nodes on the path from root to a certain node K in the tree. Write code which takes the search tree B and the sequence of keys S as input and outputs the pointer to node K (i.e., the pointer to node containing a_k as key).

Q: Write code to search for element x in a sorted array. The function you write should return 1 if x is present in the array and the corresponding array index. If x is not found, the function should return –1 and return an index value of –1.

Q: Write a recursive function which prints the keys of only the leaf nodes in a binary tree.

Q: Write code to implement operations of a Queue using a Stack ADT.

Q: Write code to implement operations of a Stack using a Queue ADT.

Q: Somehow, you did not realize your stack implementation uses Queue ADT. You did Queue using Stack as ADT (which internally uses Queue ADT). What is the complexity of Enqueue and Dequeue in this Queue implementation?

Q: Write code which takes an inorder traversal, preorder traversal sequence for a binary search tree as input and reconstructs the tree.

Q: Prove that a binary tree of N nodes will have N+1 NULL links.

Q: Show the step-by-step result of inserting 2,1,4,5,9,3,6,7 into an initially empty AVL tree. Draw the tree after each insertion (and any rotation needed). Mention which rotation is used (if a rotation is applied).

Q: Show the step-by-step result of inserting 3,1,4,6,9,2,5,7 into an initially empty binary search tree. Also show the result of deleting the root of the tree.

Q: Write code which checks whether a given tree is a valid binary search tree.

Q: Write code which checks whether two trees are 'identical'. If two trees are both empty, they are considered identical. If the two trees are not empty and have identical keys in the respective root nodes and identical left and right subtrees, they are considered identical.

Q: Write code which takes a pointer to a binary tree as input and computes

(a) the number of nodes

(b) number of leaves

Q: Prove that BuildHeap(int *Array, int n) can be done in O(n) worst case where is n is number of elements in Array.

Q: Suppose a heap with N elements is represented as an array (0 indexed). Let d be the depth of a node in the heap whose array index is k. What is the mathematical relationship between d and k?

Q: Show the tree and the array for heapsort as the algorithm processes the input. Use this sequence as example: 142,543,123,65,453,879,572,434,111,242,811,102

Q: Suppose there is an array of elements containing {-1,0,1}. Write an O(N) algorithm to rearrange the elements so that all –1s precede all 0s and all 0s precede the 1s. You should do only a single pass on the array and use constant O(1) extra space.

Q: Supose you are given an array which contains N numbers. Write a O(N^3) algorithm which determines if there are three unique elements in array whose sum equals a given number k.

Q: For the in-situ permutation (indirect sorting using pointer cycles). Let N be the number of elements being sorted. Let p be a location index in the array. Show that the probability that p is in a cycle of length 1 is 1/N.

Q: In Hash table with Quadratic Probing, prove that if table size is prime number and the table is at least half empty, we can always insert an element.

Q: You have given a HashTable h1 which uses linear probing. Your goal is to return HashTableQP h2 which uses quadratic probing. Assume that you have access to the following header files (first one is for linear probing and the second one is for quadratic probing) and the functions defined in these header files.

HashTable.h

```
struct stHT{
   int iTableSize;
   Node pStart;
};

struct stNode{
   Element iElement;
//    int iArryofNoUse[100000];
   Node pNext;
};

typedef struct stHT * HashTable;
typedef struct stNode * Node;
typedef int Element;
typedef int Key;
```

```
HashTable   CreateHashTable(int iTableSize);
void        InsertElement(Element n, HashTable myHt);
void        EmptyTable(HashTable myHt);
Key         getHash(Element n,HashTable myHt);
```

HashTable_QP.h

```
typedef struct stHtQP * HashTableQP;
typedef struct stNodeQP * NodeQP;
typedef int Element;
typedef int Key;

HashTableQP     CreateHashTableQP(int iTableSize);
void            InsertElementQP(Element n, HashTableQP myHt);
Key             getHashQP(Element n,HashTableQP myHt);

enum Type {Legitimate, Empty, Deleted};

struct stHtQP
{
   int iTableSize;
   int iNumberofElements;
   Node pStart;
};

struct stNodeQP{
   Element iElement;
   enum Type type;
};
```

Q: Given an undirected graph with n vertices, no matter how you complete DFS, I can always complete BFS in the same order. What can you say about this graph?

Q: Consider an undirected complete graph of n vertices.

a. How many DFS are possible starting at node 1?
b. How many BFS are possible starting at node 1?

Q: Construct a Graph with most of the edges having positive edge but at least one edge with negative edge but shortest path from each node to each node exists.

Q: How many edges does a forest with n vertices and k trees contain? (Prove your answer)?

Q: If there are n nodes in an AVL tree, what is minimum possible height of the tree?

Q: Show how AVL tree insertions will happen if I insert the following nodes in an empty AVL tree.

2,12,4,1,7,8,5,3,6,11

Q: Construct a five-node (Nodes:1,2,3,4,5) graph such that if you find an MST (Minimum Spanning Tree) rooted at node 1, you still can use the MST to go from 1 to all the nodes in the least cost.

Q: Construct a five-node (Nodes:1,2,3,4,5) graph such that if you find an MST (Minimum Spanning Tree) rooted at node 1, you still cannot use the MST to go from 1 to nodes 4,5 in the shortest viable way.
