Data Structures
IMT + IMG II Semester

Note:

- Lengthy paper
- For any part, if you need to re-use the code for an earlier part (with some changes), you can make a direct function call (and write the changes to the previous code, if any). Do not copy-paste a code from one part to another.

1. **Searching and Sorting.**
   a. You are given an unsorted array X and an element a. Write a C/C++ program to search element a in the array and return the index of a in X. If the element is not found, return -1. Also, write the complexity. **[2]**
   b. Write a C/C++ program to implement heap sort in an array X. Ensure to write the code for all non-standard helper functions that you may have used. Also, write the complexity. **[3.5]**
   c. Heap sort needs to be implemented on the following array [6, 4, 2, 7, 9, 3]. Show all the steps of the working of the algorithm. **[2]**
   d. Write a C/C++ code to implement binary search on a sorted array. Also, write the complexity. **[2]**
   e. You are given an unsorted array X. The user gives multiple elements a one by one from the command line. After every input, you need to give an output as the index of a in X. If the element is not found, return -1. There will be a total of q inputs from the user (where q is of the order of n). Write the C/C++ code for taking the input, performing the operation, and giving the output. Also write the complexity for all q queries. **[2]**
   f. You are given a **sorted** array X (data) and a **sorted** array Y (queries). For every element in Y (queries), return the index at which the element is found in X (data). The output would be an array storing the indices of all elements in Y (queries). Note that both the data and queries are **sorted**. There are a total of q queries. q is of the order of n. Also write the complexity for all q queries.
   Suppose if X (data)=[2, 4, 8, 9, 10] and Y (query)=[2, 2, 4, 9, 9], then the output is [<u>0</u>, <u>0</u>, <u>1</u>, <u>3</u>, <u>3</u>] because Y[0]=2 is at index <u>0</u>, Y[1]=2 is at index <u>0</u>, Y[2]=4 is at index <u>1</u>, Y[3]=9 is at index <u>3</u>, Y[4]=9 is at index <u>3</u>. The output array is given by the underlined numbers. **[3]**

2. **Towards graph search.** Suppose all students of the course have a unique firstname followed by a space followed by last name (e.g. nilanjan mitra). The names are all small case containing only letters *a-z*. An edge exists between two students who are friends with each other.
   a. Write a C/C++ program to convert string names into an integer hash-value. **[1]**
   b. Write a C/C++ code implementation of hashing by linear probing. You may assume string keys that additionally store integer values. The code should ensure that no duplicate keys get inserted. **[2]**
   c. Write a C/C++ code to add a node at the beginning of a linked list. The linked list stores string names (instead of the usual integer). **[1]**
   d. Make a C/C++ function "add edge" that adds an edge between 2 students (e.g. add an edge between "nilanjan mitra" and "udit shrivastava") using **adjacency matrix**. The vertices are strings. (*Hint*: store the adjacency matrix as a hash table). **[2]**

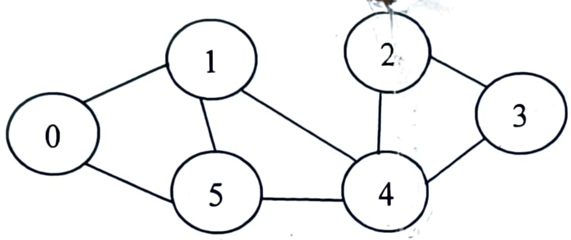
e. Make a C/C++ function "add edge" that adds an edge between 2 students (e.g. add an edge between "nilanjan mitra" and "udit shrivastava") using **adjacency list**. The vertices are strings. (*Hint:* store the adjacency list as a hash table). **[2]**

f. Write the **pseudo-code** for Breadth First Search Algorithm. If needed, assume integer vertices numbered 0 to V-1. Also, write the complexity. **[3.5]**

g. Suppose the following integers are added into a hash table of size 13 using the hash function $h(x)=x \% 13$ using hashing by linear probing. The data is [50, 81, 64, 24, 10, 76, 14]. Give the final hash table. **[2]**

h. Show the adjacency list and adjacency matrix corresponding to the following graph. **[2]**

i. Suppose the Breadth First Search algorithm is used on the following graph. Show the steps, costs, parents and final output. **[2]**



3. **Trees**

a. Suppose the following nodes are inserted in the same order in a Binary Search Tree. Give the final tree formed. Data: [5, 8, 4, 1, 2, 9, 7]. **[2]**

b. Give the pre-order, post-order and in-order traversals of the tree obtained from (a). **[3]**

c. In the tree obtained from (a), first 5 is deleted, and then 7 is deleted. Give the tree formed after both operations. **[1]**

d. Suppose the following nodes are inserted in the same order in an AVL Tree. Give the final tree formed. Data: [5, 8, 4, 1, 2, 9, 10] (note that the input is different from (a)). **[2]**

e. Write a C/C++ code to insert an element in a Binary Search Tree **[2]**

f. Consider a Binary Search Tree implementation that stores upto k elements at any leaf node. The moment an integer is inserted such that the capacity of the leaf node has to be increased to more than k, the leaf is broken down into children such that no child has more than k elements as leaves. Write the **pseudo-code** for the insert operation. **[3]**

A leaf can store man of ak element of data