**Programming Practices**

General Instructions:

All code will require

1. The pseudocode: This should explain the approach in a step-by-step fashion
2. Adequate comments: Each function should have a comment stating its purpose. Each variable of each function should have a comment that explains its purpose, with reference to the pseudocode
3. Proper structuring: The code should be divided into classes, with their member variables and functions such that the classes and functions are cohesive
4. Proper indentation: The code within a function should be properly indented so that it is readable.

Evaluation is based on the correctness of the code as well as the above components. Merely writing a code that is correct will not ensure full marks.

Week 1 (Sorting and selection)

1. Write a program that finds the k-th largest number in an unsorted array of length N, in O(N) time.
2. Write a program to generate one hundred million random numbers of type double and store it in a file. Now write a second program that will sort these numbers and store them in another file. Use a strategy that will minimize disk I/O.

Week 2 (Verification of randomness)

1. Write a program that finds the histogram of a given set of random numbers that were generated in Week 1. Find the standard deviation of the frequencies in each bin of the histogram.
2. Write a program that applies the Kolmogorov test on the set of random numbers generated in Week 1.

Week 3 (Binary (Search) Tree)

1. Write a program that inserts random numbers (between 1 and 100) into a binary search tree, where duplicate numbers are not inserted. The random numbers should not be stored in an array. The total number of nodes in the tree has to be taken as input. Store the tree so formed in a file. Read the file back and provide the inorder traversal of the tree.
2. Consider the above file that contains the binary search tree. Provide a procedure which performs deletion of a particular given value (node) from the tree. Verify that the tree after deletion of the node is still a BST.

Week 4

Evaluation based on the assignments of earlier weeks.

Week 5 (Heaps)

1. Implement priority queue using heap.
2. Write a program to perform deletion of a node from a given heap. Ensure that the heap property is preserved.

Week 6 (AVL Tree)

1. Construct an AVL tree from a given array of numbers.
2. Provide procedure to delete an input number from an AVL tree ensuring that all the properties of AVL tree are still maintained.

Week 7 (Trie)

1. Write a program to accept a set of characters along with their frequency of occurrence. Build the Huffman code for each character.
2. Assume that you are given a text file containing words in English. The words are separated by white spaces or punctuation marks (like ‘.’, ‘,’, ‘;’, , ‘?’, ‘!’). Now write a program that extracts all the words from the text file and writes each word on a separate line of a different file. Now write a separate program that will read the separated words and build a trie from these words. Now, input an arbitrary word. If the word exists in the tries then the program should print “OK”. If the word does not exist in the trie (may be due to spelling mistake) then the program should print “NOT FOUND”.

Week 8

Evaluation based on the assignments of earlier weeks.

Week 9 (Multi threading)

Important note: Pay attention to synchronization between the threads in the following two assignments.

1. Write a program has two threads. The first thread opens a text file for reading and reads one character at a time and writes the byte into a queue. If the queue is non empty then the second thread extracts a byte from the queue and writes into another file.
2. Write a program that simulates a buffer as a circular list of finite size. The program should have
   1. One thread to create a set of random numbers (the number off numbers is not fixed). If the circular list has space accommodating the numbers then the numbers are written into the list, else the thread tests the list periodically to check whether there is adequate space.
   2. The second thread reads the numbers from the list, removes them from the list. and displays them on the screen.

Week 10 (Graphs)

1. Write a program reads the names of some stations from a file. It then reads the names of some trains. Corresponding to each train it reads a list of stations that this train stops at. Now your program should accept a pair of stations as input. If there are any direct trains between the given pair of stations then your program should print the names of these trains. In case there is no direct train between the given pair of stations then the program should try to find a pair of trains such that a passenger can travel between the given pair of stations by changing from one train to the other at an intermediate station. If such a pair of stations is found then the program should print the names of the pair of trains and also give information about the station where the change over is required.
2. Write a program that finds all distinct paths (along with their weights) in a given weighted graph and for a given pair of vertices. Ensure that you do not get stuck in a cycle.

Week 11

Revision and catching up.

Week 12

Evaluation based on the assignments of earlier weeks.