Programming Assignment: 1

Deadline: 21 August, 2023 11:59 PM **[Maximum Marks: 20]**

Submission Policy and Requirements :

- Programming languages: C
- Do cite references (if using any)
- Submissions should include a working code for the questions asked, a report to show the analysis of results in each part
- Submission of the report is mandatory.

Assessment criterion:

The assessment will be done on the basis of the following components:

- Working codes
- Analysis and clarity of results & clarity of the report
- Understanding the theoretical concepts

Guidelines for Submission:

- A single report(pdf) for all questions
- Mention all the relevant results and comparisons as asked or wherever required for a better understanding of the results
- A single zip file containing the report, codes
- Name the file with the roll number, for example, Roll_Number_PA1.zip

Question 1: **Marks: 10**

Title: Efficient File System Indexing: Performance Analysis

Introduction:

Rutvik is a software engineer. Rutvik and his team are working on optimizing a file system for a cutting-edge operating system called NextOS," and you are also a part of his team. NextOS is designed to handle massive amounts of data and provide lightning-fast file access. One of the key challenges your team is facing is optimizing the file system to efficiently store and retrieve files and directories.

The file system currently employs a basic data structure for indexing files and directories, but your team believes that implementing a more advanced data structure could significantly improve the performance of file insertion, searching and deleting operations. Rutvik gives you a task to compare two different data structures: your own custom data structure (the current existing data structure other than the red-black tree) and a red-black tree for this task.

Task:

Your task is to implement two different data structures for file system storage (insertion), retrieval (searching) and deleting. One data structure is your own choice, which is called a custom data structure, and the second is a red-black tree.

Once you have both data structures implemented, you need to compare their performance in terms of the time taken for insertion, searching and deleting operations.

Input:

For the first line of input, you have to construct both data structures.

For the second line of input, you have to provide a set of keys. Now you have to insert those keys into both data structures.

For the third line of input, you have to retrieve those keys from both data structures. For those keys that are not found, you have to return -1.

Fourth line indicates to delete these keys.

https://docs.google.com/document/d/1NptZ6gA70YGm8ulbv36YJk9zzNvW6d9qYxoE9Md0Yf4/edit?usp=sharing

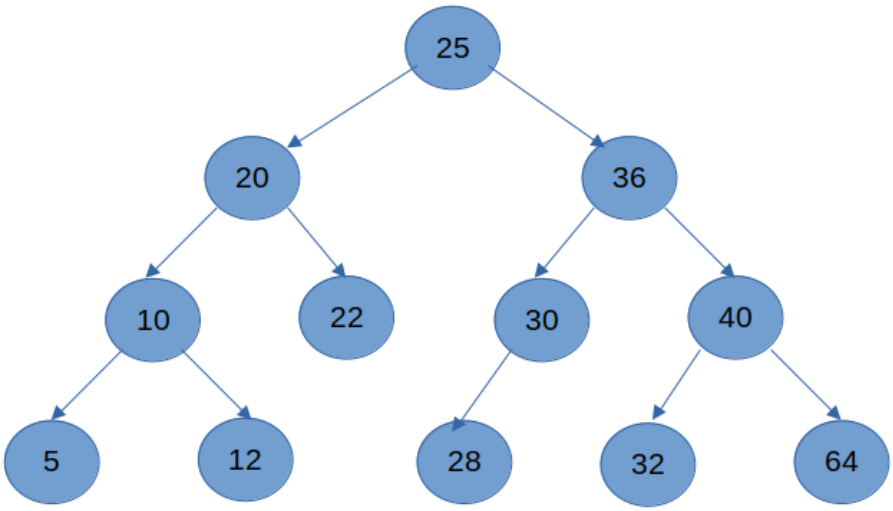
Output:

For performance analysis, you have to calculate the time taken by your algorithm and the red-black tree for the storage (insertion), retrieval (searching) and deleting of the provided keys of the input lines of second and third with proper explanation.

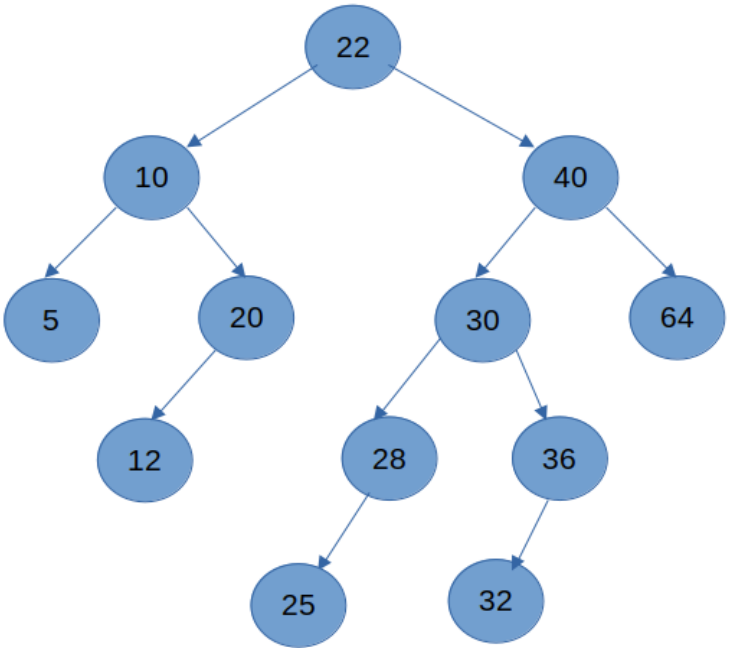
Question 2: **Marks: 05**

write a code to convert an arbitrary n-node binary search tree to a given n-node binary search tree.

Input:



Output:



Question 3: **Marks: 05**

implement Dijkstra's algorithm using Red Black Tree.