Cairo University
Faculty of Computers & Artificial Intelligence.
Algorithms Analysis & Design
Third Year
Dr. Hanaa & Dr. Zeinab
2022/2023

Assignment #2

Question 1 (5 Grades)

We need to build a server that can store secret notes for users. The server can store/show/delete the secret notes when the user enters his credentials. You can store the users' credentials (username and password) in plaintext on the server. So, when the user enters the username and password, the server retrieves the user's notes and prints it.

Storing the user's credentials as plain text is not a secure strategy. If someone could hack the server, it could easily find the stored credentials and unravel all the user's secret notes. One better strategy is to encrypt the credentials using the hashing and store the hashed credentials instead of the plaintext credentials.

You are required to build a hash function that takes the credentials and converts it to a hash index which represents the user's name and password.

Construct a server (class) that has the following:

- **credentials_file_path**: string; where this file contains the hash_index, secret_notes_file_path for each secret note.
- **table**: array of strings; is a hash table that contains the hashed credentials along with the file path to the user's secret notes
- **table_size**: int; it is the size of the table which is equal to 1 000 000 007
- **server(credentials_file_path: string)** ; is a constructor that creates credentials file if not existed or loads the credentials file into the hash table
- **~server()**; is a destructor that stores the latest hash table in the credentials file
- **load_credentials()**; this function loads the credentials into the table from the credentials_file_path. (0.5)
- **store_credentials()**; this function stores the table that is filled with the credentials into credentials file. (0.5)
- **hash(username: string, password: string)**: int; this function takes the credentials and outputs the hashed index in the table using the following equation: (1)

$$h(S) = \left(\sum_{i=0}^{|S|-1} S[i]x^i \right) \bmod m,$$

where $S[i]$ is the ASCII code of the i-th symbol of S, $m = 1\ 000\ 000\ 007$ and $x = 263$.

- **insert_secret_notes(username: string, password: string, secretNotes: string)**; this function stores the secret notes file path in the table. The secret notes file is generated. The file name could be user's_hashedCode.txt

  - **Example:** User's credentials hash code is 12345 so, the filename is 12345.txt

- **show_secret_notes(username: string, password: string)**; it prints the secrets on the console if the username and password are found in the table. Otherwise, it prints **"Invalid credentials"**

- **delete_secret_notes (username: string, password: string)**; this function deletes the record from the table if the username and password are correct and prints to **"The user Secrets have been deleted"**, otherwise prints **"Failed operation"**.

- Implement the **main** function of the program to show a menu to the user that allows him to choose an action to perform and takes the necessary input from him. After performing the action, the user will be shown the same menu again.

  Example for the menu:

```
Please choose an action to continue:
1. Insert secret notes   // user inputs the credentials, and secret notes (1)
2. Show secret notes   //  user inputs the credentials (1)
3. Delete secret notes  //  user inputs the credentials (1)
```

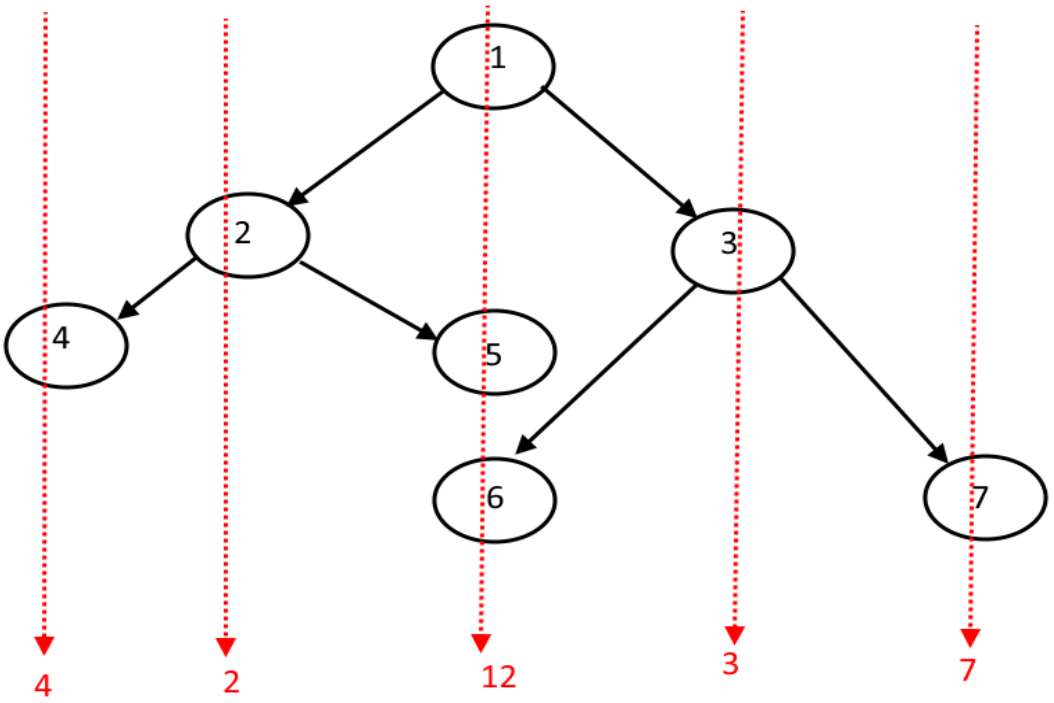
Question 2 (5 Grades)

A **Red–Black Tree** is a kind of self-balancing binary search tree. Each node of the binary tree has an **extra bit**, and that bit is often interpreted as the color **(red or black)** of the node. These color bits are used to ensure the tree remains approximately balanced during insertions and deletions

.

In this task, you are required to develop a **Generic Red Black Tree** using **C++** with the following specifications and functions:

- The tree must use **C++ templates** to accept any data type
- The tree will contain **unique values** only
- The tree class will contain the following functions as public functions:
  - **void insert(T value)** // inserts a new value into the tree while keeping it balance
  - **void getHeight()** // returns the maximum height of the tree
  - **T uncle(T* value)** // returns the uncle of a given node that contains a certain value
  - **bool isBalanced(T node)** // returns true if the given subtree is balanced
  - **bool search(T value)** // returns true if a given value exists and prints the path to reach that node
  - **void print()** // displays a visual readable print of the tree nodes including colors
  - **void printVerticalSum()** // prints the sum of each column of the tree (see next figure)



- You can add any other functions that will assist you with completing the implementation of the previous functions in the private section of the class
- Implement the main function of the program to show a menu to the user that allows him to choose an action to perform and takes the necessary input from him. After performing the action, the user will be shown the same menu again.
  Example for the menu:

Please choose an action to continue:
1. Insert value // user inputs a value (1)
2. Display height // the height is displayed (0.5)
3. Display uncle of value // user inputs a value (0.5)
4. Check if subtree is balanced // user inputs the root value of the subtree (1)
5. Search for value // user inputs value (0.5)
6. Print tree (0.5)
7. Print vertical sum (1)

Submission instructions:

- **The assignment is submitted in a group of maximum 4 students.**
- **Submission on classroom Due date 17/12/2022**
- **Submission file should have the name of your IDs for example:**
  - **Group_ID1_ID2_ID3.zip**