National Institute of Technology Calicut
Department of Computer Science and Engineering
Fourth Semester B. Tech (CSE)-Winter 2021-22
CS2094D Data Structures Laboratory
Assignment #2

Submission deadline (on or before): 07.02.2022, 9:00 AM

Policies for Submission and Evaluation:

- You must submit your assignment in the Eduserver course page, on or before the submission deadline.
- Ensure that your programs will compile and execute without errors using gcc compiler.
- During the evaluation, failure to execute programs without compilation errors may lead to zero marks for that evaluation.
- Your submission will also be tested for plagiarism, by automated tools. In case your code fails to pass the test, you will be straightaway awarded zero marks for this assignment and considered by the examiner for awarding F grade in the course. Detection of ANY malpractice related to the lab course can lead to awarding an F grade in the course.

Naming Conventions for Submission

- Submit a single ZIP (.zip) file (do not submit in any other archived formats like .rar, .tar, .gz). The name of this file must be

  `ASSG<NUMBER>_<ROLLNO>_<FIRST-NAME>.zip`

  (Example: *ASSG*1_*BxxyyyyCS_LAXMAN.zip*). DO NOT add any other files (like temporary files, input files, etc.) except your source code, into the zip archive.

- The source codes must be named as

  `ASSG<NUMBER>_<ROLLNO>_<FIRST-NAME>_<PROGRAM-NUMBER>.c`

  (For example: *ASSG*1_*BxxyyyyCS_LAXMAN*_1.*c*). If you do not conform to the above naming conventions, your submission might not be recognized by our automated tools, and hence will lead to a score of 0 marks for the submission. So, make sure that you follow the naming conventions.

Standard of Conduct

- Violation of academic integrity will be severely penalized. Each student is expected to adhere to high standards of ethical conduct, especially those related to cheating and plagiarism. Any submitted work MUST BE an individual effort. Any academic dishonesty will result in zero marks in the corresponding exam or evaluation and will be reported to the department council for record keeping and for permission to assign F grade in the course. The department policy on academic integrity can be found at: http://cse.nitc.ac.in/sites/default/files/Academic-Integrity_new.pdf.

QUESTIONS

1. Write a program to implement a HASH TABLE data structure to store the Employee details with *employee_id* as key k. Your program should contain the following functions:

   - HASHTABLE(m): Creates a hash table T of size m.
   - INSERT(T,k): Inserts an element into hash table T having key value k.
   - SEARCH(T,k): Checks whether an element with key 'k' is present in hash table T or not.
   - DELETE(T,k): Deletes the element with key 'k' from hash table.
     Note: Assume that the deletion operation will always be a valid operation. i.e. the element to be deleted is present in the hash table.

   Input Format:

   - The first line contains a character from { 'a', 'b' }:
     - Character 'a' denotes the collision resolution by Quadratic Probing with hash function
       $$h(k,i) = (h_1(k) + C_1 i + C_2 i^2)\ mod\ m$$
       where $h_1(k) = k\ mod\ m$, C_1 and C_2 are positive auxiliary constants and $i \in [0, m-1]$.
     - Character 'b' denotes collision resolution by Double Hashing with hash function
       $$h(k,i) = (h_1(k) + i * h_2(k))\ mod\ m$$
       where $h_1(k) = k\ mod\ m$, $h_2(k) = R - (k\ mod\ R)$ { R = the largest Prime number less than the size of hash table } and $i \in [0, m-1]$.
   - Second line contains an integer $m \in [1, 100]$, denotes the size of the hash table.
   - In case of quadratic probing (option a), next line contains two integers C_1 and C_2 separated by space.
   - Subsequent lines may contain a character from { 'i', 's', 'd', 'p', 't' } followed by zero or one integer.
     - i k: insert the element with key k into hash table
     - s k: search the element with key k in hash table. If the key is present in hash table, then print 1. Otherwise, print -1.
     - d k: delete the element with key k from hash table.
     - p: print the hash table in "index (key values)" pattern. If no key values are present in an index, then print "()" after "index" (Refer sample output for explanation).
     - t: terminates the program

     Note: Total number of elements n to be inserted into hash table will be less than or equal to size of hash table m i.e., $n \leq m$.

   Output Format:

   - The output (if any) of each command should be printed on a separate line.

   Sample Input 1:

   ```
   a
   7
   0 1
   i 76
   i 40
   i 48
   i 5
   s 5
   i 55
   p
   s 62
   d 55
   t
   ```

Sample Output 1:

1
0 (48)
1 ()
2 (5)
3 (55)
4 ()
5 (40)
6 (76)
-1

Sample Input 2:

b
7
i 76
i 93
i 40
i 47
i 10
i 55
p
d 40
s 47
s 76
s 40
t

Sample Output 2:

0 ()
1 (47)
2 (93)
3 (10)
4 (55)
5 (40)
6 (76)
1
1
-1

2. Write a program to group the words according to their lengths from a given string S using a HASH TABLE of size k with separate chaining. Assume that only alphabets are present in the string S and maximum size of the string is 500. The words of the string are grouped using the following formula.

$$Index\_No = (length\_of\_word \times length\_of\_word) \ \% \ k$$

where % is the modulo operation and k is the size of hash table. If the string contains multiple occurrences of a word w, then it should not be added again in a group. Only the first occurrence of w is added to the group.

Note: Hash table is implemented as an array in which each entry contains a head pointer to a linked list which contains the words of the same group. Words generating same *Index_No* belong to the same linked list (refer sample output for explanation). Duplicate words are not allowed in the list. Each node of the linked list is of the following type.

```
struct node{
  char *word; // word to be store
  struct node *next; //pointer to the next node
};
```

Input Format:

- First line of the input contains an integer 'k', the size of the hash table.
- Second line of the input contains a string/sentence of words.

Output Format:

- Each line of the output should print the index number and words in it, separated by a colon(:).
- The words inside a group are separated by minus sign(-).
- If no words are present in the group then print 'null' in place of words.

Sample Input 1:

3
Write a program to create a hash table

Sample Output 1:

0:create
1:Write-a-program-to-hash-table
2:null

Sample Input 2:

5
This program is a program to create a hash table

Sample Output 2:

0:table
1:This-a-create-hash
2:null
3:null
4:program-is-to

3. The Ministry of Health and Family Welfare is collecting the details of people who suffer from rare diseases and their relatives in order to identify the effects of heredity. The Ministry uses a Hash Table-Binary Search Tree hybrid data structure H for organizing the data (see Figure 1). The last name of a person is used as a key 'k' to hash into the table of size 128. The entries hashed into the same bucket can be considered as *relatives* (even if they don't share the same last name) and are organized as a BST using age. The hash function $h(k)$ to be used is as follows:

$$h(k) = (\texttt{sum of the ASCII values of each character in the key k})\%128$$

Given a list of full names (firstName lastName) and ages, write a C program to organize it into the data structure H described above. Since many rare diseases are hereditary, it is also necessary to identify the relatives of people who suffer from such diseases. Given the full name of a person P, your program should also print the full names and ages of all people in the path from the root to P, in the BST that contains P. Your program should contain the following functions:

- InsertData (string *firstName*, string *lastName*, int *age*): Applies the hash function $h(k)$ to *lastName* to obtain the pointer to the root of a BST. Inside this BST, a node containing the details $\langle$*firstName*, *lastName*, *age*$\rangle$ is inserted using *age* as key.
- PrintRelatives (string *name*): print the full names and ages of all people in the path from root to the person named *name*, in the BST that contains *name*. If an entry corresponding to the *name* does not exist in the data structure H, print -1.

Input Format:

- The input contains multiple lines, each line starting with a character from { 'i', 'p', 't'}:

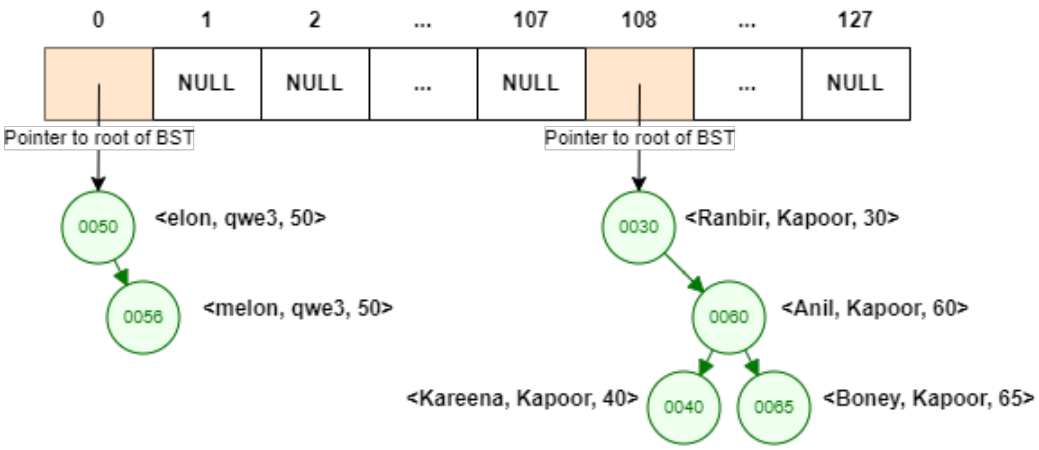

Figure 1: The data structure after performing the operations in the Sample Input of Q3.

- Character 'i' is followed by two alphanumeric strings and an integer $\in [1, 100]$, denoting *firstName*, *lastName* and *age*, respectively. In this case, your program should insert these details into the data structure by calling the InsertData (string *firstName*, string *lastName*, int *age*) function.
- Character 'p' is followed by two alphanumeric strings denoting *firstName* and *lastName*, respectively. In this case, your program should print the full names and ages of all people in the path from root to the person named *firstName lastName*, in the BST that contains *firstName lastName*, by calling the PrintRelatives (string *name*) function.
- Character 't' terminates the program.

Output Format:

- For input lines starting with character 'p', each $\langle$*full name*, *age*$\rangle$ entry in the path must be printed on a separate line.

Sample Input:

```
i Ranbir Kapoor 30
i Anil Kapoor 60
i Kareena Kapoor 40
i elon qwe3 50
i Boney Kapoor 65
i melon qwe3 56
p Boney Kapoor
p melon qwe3
p Sonam Kapoor
p Sachin Tendulkar
t
```

Sample Output:

```
Ranbir Kapoor 30
Anil Kapoor 60
Boney Kapoor 65
elon qwe3 50
melon qwe3 56
-1
-1
```

4. Suppose you're in the data organizing committee of your institute. Your task is to assign a unique

roll number to every student in your institute using the name of the student and store the details into a Hash table H of size 1000. Roll number contains a unique alphabet followed by a numeric id of *three* digits which can be calculated with the help of a hash function $h(k)$. The first letter of the roll number is the first letter of the student *name* and the last three digits are the hash numbers generated by the hash function given below.

$$h_i(k) = ((\texttt{sum of the ASCII values of each character in the key k})\%26)\%10$$

where $h_i(k)$ is the i^{th} digit in the *roll_number* for $1 \leq i \leq 3$ and the key k is identified in the following way:

- For every digit formation, the key k is generated by selecting at most *three* (if possible) characters from the student *name* starting from the left side.
- To find the key k for the:
  - **first digit**; select 0^{th}, 1^{st}, and 2^{nd} character from the *name*.
  - **second digit**; select 0^{th}, 2^{nd}, and 4^{th} character from the *name*.
  - **third digit**; select 0^{th}, 4^{th}, and 8^{th} character from the *name*.
- In the i^{th} digit formation, if there are only less than three characters available, then use only that available characters in such cases.

i.e., $roll\_number =< first\_letter\_of\_your\_name > h_1(k)h_2(k)h_3(k)$. Now the student *name* is inserted into the index $h_1(k)h_2(k)h_3(k)$ of H.

For example, **Abhishek** is the given name then **'Abh'** are the letters chosen for 1^{st} digit formation, **'Ahs'** are the letters chosen for 2^{nd} digit formation and **'As'** are the letters chosen for 3^{rd} digit formation. In the above example, for the 3^{rd} digit formation only two characters are available. So use only these two available characters in this case.

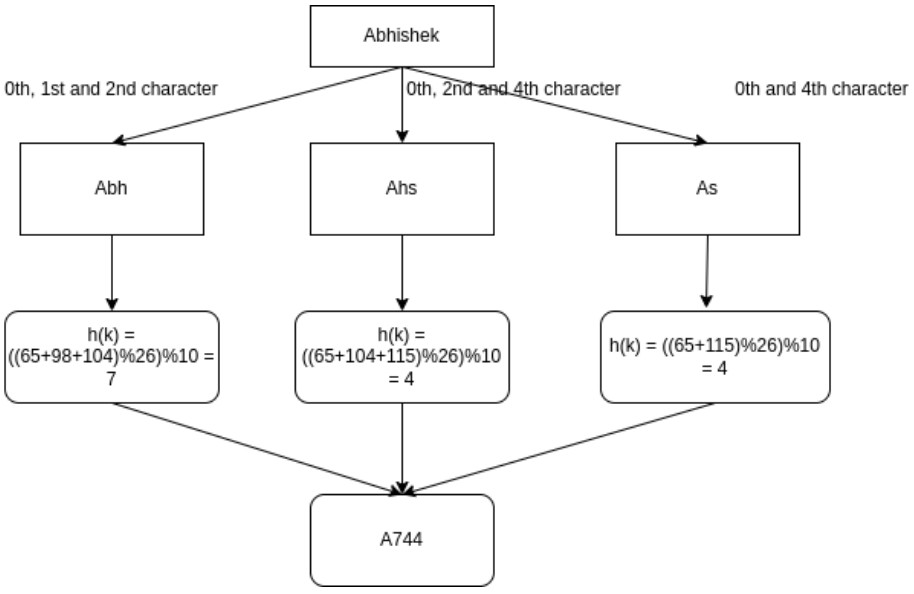

Figure 2: Formation of roll number for a given example Abhishek

$$h(Abh) = ((65 + 98 + 104)\%26)\%10 = 7$$
$$h(Ahs) = ((65 + 104 + 115)\%26)\%10 = 4$$
$$h(As) = ((65 + 115)\%26)\%10 = 4$$

Thus, the roll number associated with Abhishek is A744. Now the name Abhishek is inserted into the index 744 of H.

Your program should contain the following functions:

- InsertData (string *Name*): Inserts *Name* into the index x of the Hash table H. This function first applies the hash function $h_i(k)$ to obtain the roll number associated with the *Name*, then the three digits in that generated roll number is the index x.
- Search (string *RollNum*): Prints the name of the person associated with the user input *RollNum*.
- Delete (string *RollNum*): Deletes the name of the person associated with the user input roll number from the hash table.

Input Format:

- The input contains multiple lines, each line starting with a character from { 'i', 's', 'd', 't'}:
- Character 'i' is followed by an alphanumeric string, denoting the *name*. In this case, your program should insert the *name* into the index obtained from the RollNumber calculated using the hash function.
- Character 's' is followed by an alphanumeric string, denoting the *RollNum*. In this case, your program should print the name of the student in the index obtained from the *RollNum* in H. If the entry corresponding to the index obtained from *RollNum* is empty, print NOT FOUND.
- Character 'd' is followed by an alphanumeric string, denoting the *RollNum*. In this case, your program should delete the name of the student in the index obtained from the *RollNum* in H.
- Character 't' terminates the program.

Output Format:

- The output (if any) of each command should be printed on a separate line.

Note: Assume that the Hash table H is free from collision and the deletion operation is always a valid operation.

Sample Input:

```
i Abhishek
i Anupam
i Akshat
i Ebin
s A744
s E287
s A333
d A744
s A744
t
```

Sample Input:

```
Abhishek
Ebin
NOT FOUND
NOT FOUND
```