Department of Computer Science and Engineering

National Institute of Technology Calicut
CS2094D DATA STRUCTURES LABORATORY
(Second Semester MCA)

ASSIGNMENT 2

General Instructions

1. **Try to complete all the assignment questions and submit in eduserver platform**
2. **Your assignment is due by 11.30pm on 31/03/2022**
3. **All programs should be implemented in C language**
4. **You have to upload the complete code as a zip file named as Firstname-Regno-Assg#(e.g., John-m200034ca-Assg1. Zip)**
5. **Students who submit works that are found to have been plagiarized can be cause for an automatic 0 marks**
6. **All submitted code should be well-documented, and the code should be easily understood by reading the comments**

Questions

1. Write a C program to implement hash table data structure using open addressing to store student's information with roll number as key.
Your program should contain the following functions:-
- hashTable(int m)- create a hash table of size m
- insert(int k)- insert element into hash table having key value as' k '
- search(int k)- find whether element with key 'k' is present in hash table or not
- delete(int k)- delete the element with key 'k'

Input Format:

The first line contains a character from { 'a', 'b', 'c', 'd'} denoting

a- Collision resolution by Linear probing with hash function

$h(k,i) = (h_1(k) + i)$ mod m where $h_1(k) =$ k mod m

b - 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, i = 0,1,...m-1

c - Collision resolution by Double Hashing with hash functions

$h(k,i) = (h_1(k) + i * h_2(k))$ mod m

Where, $h_1(k) =$ k mod m , $h_2(k) =$ R - (k mod R) { R = Prime number just smaller than the size of table }

The second line contains an integer, m, denoting the size of the hash table.
In case of quadratic probing only (option b) Next line contains the constants c1 and c2 separated by space

Next lines contain a character from { ‘i’, ‘s’, ‘d’, ‘p’, ‘t’ } followed by zero or one integer.
i x - insert the element with key x into hash table
s x - search the element with key x in the hash table.
Print 1 if present otherwise print -1
d x - delete the element with key x from the hash table.
p - print the hash table in “index (key values)” pattern.(See sample output for explanation)
t - terminate the program

(Note : In case of Linear probing, quadratic probing and Double hashing, total elements (n) to be inserted into hash table will be lesser than or equal to the size of the hash table (m) i.e. n<=m)

Output File Format:
The output (if any) of each command should be printed on a separate line

Sample Input and Output

Input 1:
```
a
7
i 76
i 93
i 40
i 47
i 10
i 55
p
s 35
s 47
d 47
s 55
T
```

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

Input 2:
```
b
7
0 1
i 76
i 40
i 47
i 5
s 5
i 55
p
s 62
```

Output 2:
```
1
0 (5)
1 ()
2 (47)
3 (55)
4 ()
5 (40)
6 (76)
-1
```

Input 3:
```
c
```

```
7
i 76
i 93
i 40
i 47
i 10
i 55
p
d 40
s 47
s 76
s 40
T
```

Output 3:

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

2. Confidential information has to be shared in the form of sentences between two space stations. The information is encrypted using a hashing technique that stores the individual words in the message on a hash table of size n with separate chaining. For each word in the sentence, two hash functions are applied separately to obtain two different positions, and the corresponding word is then stored in one of these two positions using the following two rules:

- The word is stored in the hash table entry that has a lesser number of words in the chain.
- If both the chains are of the same length then the word is added to the chain in the position calculated by the hash function *h(w)*.

The two hash functions used for a word *w* are defined as follows:

1. $$h(w) = \begin{cases} (2 * ASCII\ of\ w)\%n, & if\ length(w) = 1 \\ \lfloor(sum\ of\ ASCII\ values\ of\ first\ two\ letters\ and\ last\ two\ letters\ in\ w)/length(w)\rfloor\%n, & otherwise \end{cases}$$

2. $$g(w) = \begin{cases} \lceil(sum\ of\ ASCII\ values\ of\ all\ the\ letters\ in\ w)/length(w)\rceil\%n \end{cases}$$

Your program should implement the following functions as per the given function prototypes:

- *store_word(H, n, w):* Store the word *w* in the hash table H of size n.
- *find_hash1(w, n):* Find and return the position of *w* in the hash table of size n using the hash function *h(w)*.
- *find_hash2(w, n):* Find and return the position of *w* in the hash table of size n using the hash function *g(w)*.
- *print_table(H, n):* In a separate line, print all the words stored in each chain in the order they were inserted in the hash table H of size n, separated by a space. If there are no words in the chain at a particular position in H, print NULL.

Input Format:

- First line of the input contains an integer n, the size of the hash table.
- Second line of the input contains the message of maximum length 105 with each word having a maximum length of 100.

Output Format:

- Each of the n lines of the output should print the words stored in each chain in the hash table, separated by a space.
- If no words are present in the chain at a particular position, print NULL.

<u>Sample Input and Output</u>

Input 1:

3

First solve the problem Then write the Code

Output 1:

```
First the write

Then Code

solve problem the
```

Input 2:

```
6

Southern war field deployed planC with soldiers of team SIX
```

Output 2:

```
NULL

planC soldiers

war deployed

with of SIX

Southern team

field
```

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

$$Group\ No = (length\ of\ word * length\ of\ word)\%k$$

where *%* is the modulo operation and *k* is the size of the 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 that contains the words of the same group. Words generating the same index number belong to the same linked list. Duplicate words are not allowed in the list. Each node of the linked list should be 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 group 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
```

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
```

Output 2:

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