K. J. Somaiya College of Engineering, Mumbai-77
(A constituent College of Somaiya Vidyavihar University)

Batch: A3	**Roll. No.: 16010121045**
Experiment: 9	
Grade: AA / AB / BB / BC / CC / CD /DD	

Title: Implementation of Dictionary using map

Objective: To understand dictionary as a data structure

Expected Outcome of Experiment:

CO	Outcome
CO2	Describe concepts of advance data structures like set, map & dictionary.

Websites/books referred:

1. Michael T. Goodrich, Roberto Tamassia, and David M. Mount. 2009. Data Structures and Algorithms in C++ (2nd. ed.). Wiley Publishing

Abstract:

Dictionary ADT functions: (list and give one line details)

1. Dictionary create() creates empty dictionary

2. boolean isEmpty(Dictionary d) tells whether the dictionary d is empty

3. put(Dictionary d, Key k, Value v) associates key k with a value v;

4. if key k already presents in the dictionary old value is replaced by v

5. Value get(Dictionary d, Key k) returns a value, associated with key k or null, if dictionary contains no such key

6. remove(Dictionary d, Key k) removes key k and associated value

7. destroy(Dictionary d) destroys dictionary d

Problem statement:
Given names and phone numbers, assemble a phone book that maps friends' names to their respective phone numbers. You will then be given an unknown number of names to query your phone book for. For each queried, print the associated entry from your phone book on a new line in the form name=phoneNumber; if an entry for is not found, print Not found instead.

Note: Your phone book should be a Dictionary/Map/HashMap data structure.

Input Format

The first line contains an integer, , denoting the number of entries in the phone book. Each of the subsequent lines describes an entry in the form of space-separated values on a single line. The first value is a friend's name, and the second value is an -digit phone number.
After the lines of phone book entries, there are *an unknown number of lines of queries*. Each line (query) contains a name to look up, and you must continue reading lines until there is no more input.

Output Format

On a new line for each query, print Not found if the name has no corresponding entry in the phone book; otherwise, print the full and in the format name=phoneNumber.

Sourcecode and output screenshots:

(Students could implement map using singly or doubly linked list. It should handle all possible boundary conditions)

```
   #include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <strings.h>

typedef struct pair
{
    char *first;
    char *second;
    struct pair *next;
} pair;

typedef struct dict
{
    int size;
    pair **table;
} dict;

unsigned int hash(char *s)
{
    unsigned int hashval = 1337;
    for (int i = 0; i < (int)strlen(s); i++)
    {
        hashval = hashval * s[i] + 0xdeadbeef;
        hashval %= 0x3f3f3f3f;
    }
    return hashval;
}

void setsize(dict *d, int s)
{
    d->table = malloc(s * sizeof(pair *));
    d->size = s;
    bzero(d->table, s * sizeof(pair *));
```

```
}

void insert(dict *d, char *k, char *v)
{
    pair *p = malloc(sizeof(pair));

    char *s = malloc(strlen(k) * sizeof(char) + 1);
    char *t = malloc(strlen(v) * sizeof(char) + 1);
    strcpy(s, k);
    strcpy(t, v);

    p->first = s;
    p->second = t;
    p->next = NULL;

    unsigned int hashval = hash(k);
    if (d->table[hashval % d->size] == NULL)
        d->table[hashval % d->size] = p;
    else
    {
        pair *q = d->table[hashval % d->size];
        while (q->next)
            q = q->next;
        q->next = p;
    }
}

char *retreive(dict *d, char *k)
{
    unsigned int hashval = hash(k);
    pair *p = d->table[hashval % d->size];

    if (!p)
        return NULL;

    while (strcmp(p->first, k) != 0)
    {
```

```
        if (p->next)
            p = p->next;
        else
            return NULL;
    }

    return p->second;
}

int main(void)
{
    dict *d = malloc(sizeof(dict));
    setsize(d, 10000007);

    int T;
    scanf("%d", &T);

    char s[37];
    char t[37];
    while (T--)
    {
        scanf("%s %s", s, t);
        insert(d, s, t);
    }

    while (scanf("%s", s) != EOF)
    {
        if (retreive(d, s) != NULL)
            printf("%s=%s\n", s, retreive(d, s));
        else
            printf("Not found\n");
    }

    return 0;
}
```

Output

```
pargat@Router Code % cd
/DS/Code/"exp9
3
pargat 9912222
rahul 11122222
raju 12299933
pargat
pargat=9912222
meet
Not found
raju
raju=12299933
rahul
rahul=11122222
```

Conclusion: Successfully implemented and executed the given problem statement.

Postlab questions:

1. Give and explain at least one scenario each in which map, dictionary and set would be the most appropriate data structure to use.

Map Data Type:
Map data type is ideal to use in look-up type situations where there is an identifying value and an actual value that is represented by the identifying value. A few examples where the map data type can be used are:

- Student ID numbers and last names
- House numbers on a street and the number of pets in each house
- Postal codes and the names of cities
- Driving licenses and last names

Dictionary Data Type:
Dictionary is used in scenarios wherein we need to store an alternative value alongside with the main value. The famous example being that of storing frequency of each character alongside with the character.

- Letter and their frequency

- Key Value pairs
- Team point table

Set Data Type:
As the name defines the set data type is generally used to check whether the given elements would be considered a mathematical set or not.

- To remove repeating numbers.
- Unique elements
- Sorting unique elementds

2. State differences and similarities in set, map and dictionary.

Map	Set	Dictionary
Map stores key value pairs in it and no two values have the same keys but the different keys can store similar values. The map stores keys in sorted order.	Set data structure is also non-homogeneous data structure but stores in single row	Dictionary is also a non-homogeneous data structure which stores key value pairs
Map doesn't allow duplicate elements.	Set will not allow duplicate elements	Dictionary doesn't allow duplicate keys.
Maps can use nested among all	Set can use nested among all	Dictionary can use nested among all
Map is ordered	Set is unordered	Dictionary is ordered (Python 3.7 and above)