Computer Engineering Department

TED University

TED UNIVERSITY

CMPE 252 – C Programming, Spring 2021

Lab 04

In this lab, you are asked to complete **similars_lab4.c** program file which has been already given in Moodle. In this program, there are four functions, namely, `main`, `hash_text`, strong_similar, and `weak_similar`. `main` function is already provided, and it is supposed to remain as it is (you should not change it). You are required to implement `hash_text`, strong_similar, and `weak_similar` functions. Although this lab is very similar to the previous lab, in this time, you will solve your problem **using structures**.

There are a few global declarations in the program which are listed as below:

- Definitions of constant macros as follows: STR_LEN, MAX_ELEM, and HASHSIZE.
- A structure defined with several elements to keep the name, last name, name_lastname, and strong hash and weak hash values. The name of the defined type is Student. The hash values, and all other critical information is kept as a record by using the structure of the type of Student.

Here are the operations performed in `main` function:

- An array of records in the type of Student is created and initialized to some values. Although the hash values are set to zero, the names are initialized.
- In the for loop, the name - lastname pairs are concatenated and the hash values for strong similarity are computed.
- strong_similar function is called to find the same name-lastname pairs and fill the two-dimensional array called same to keep the mapping of similarities.
- The name-lastname pairs, the calculated hash values and similarity mapping is printed on the standard output.
- weak_similar function is called to calculate the hash values of each name and surname separately, then build the weak similarity mapping using hash values calculated.
- The name - lastname pairs, and weak similarity mapping is printed on the standard output.

Task 1: Implement `hash_text` function.

```
unsigned int hash_text (char * list) ;
```

A character pointer holding names and surnames is sent as an input, and the hash value of the input text is returned. During the calculation a basic formula is used as follows when the name is ali. The ascii code for a is 97, for l is 108 and for i is 105. The function calculates the hash value by the following formula;

hash = 26^2*97+26*108+105

The algorithm should be applied to all symbols including whitespaces between name and surname. For each symbol mod 1000 of hash value should be computed. At the end hash value should be returned.

Consider the below example run:

```
elements with hash values

ahmet yuksel        978
mehmet arslan        88
mustafa kemal       482
ali kemal          314
mustafa kemal       482
mustafa kemal       482
mehmet arslan        88
kemal ahmet         506
ali kaan           428
kemal kaan           740
```

Task 2: Implement `strong_similar` function.

```
void strong_similar (Student studentList[MAX_ELEM], unsigned int map[MAX_ELEM][MAX_ELEM])
```

Calculated hash values of all names-lastname pairs are sent to the function. The function modifies and returns the map of exact matches. Exact matching of the entries is calculated only by using hashed values.

Consider the below example run:

```
elements with hash values and similarities:

ahmet yuksel        978
mehmet arslan        88   6
mustafa kemal       482   4   5
ali kemal          314
mustafa kemal       482   2   5
mustafa kemal       482   2   4
mehmet arslan        88   1
kemal ahmet         506
ali kaan           428
kemal kaan           740
```

The matches are calculated and printed through the main program by calling the strong_similar function. Once the strong_similar function is called in main function, the above output is printed.

Task 3: Implement `weak_similar` function.

```
void weak_similar (Student studentList[MAX_ELEM], unsigned int map[MAX_ELEM][MAX_ELEM])
```

Array of structure of the type of Student is sent as input and previously calculated similarity map is sent as input-output parameter.

The function calculates hash values of names and lastnames for each pair separately by calling hash_text function. Once the hash values of all names and lastanmes are calculated the function updates the map array to mark the weak similarities (either names or surnames are same). For example;

ali kemal and mustafa kemal are weakly similar. They both are weakly similar to kemal ahmet.

Consider the below example run:

```
elements with weak similarities:

ahmet yuksel        978  7
mehmet arslan        88
mustafa kemal       482  3  7  9
ali kemal          314  2  4  5  7  8  9
mustafa kemal       482  3  7  9
mustafa kemal       482  3  7  9
mehmet arslan        88
kemal ahmet         506  0  2  3  4  5  9
ali kaan           428  3  9
kemal kaan           740  2  3  4  5  7  8
```

Once the weakly_similar function is called in main function, the above output is printed.

Sample Run:

See the next page for sample run.


```
elements with hash values and similarities:

ahmet yuksel            978
mehmet arslan            88   6
mustafa kemal           482   4   5
ali kemal               314
mustafa kemal           482   2   5
mustafa kemal           482   2   4
mehmet arslan            88   1
kemal ahmet             506
ali kaan                428
kemal kaan              740


elements with weak similarities:
ahmet yuksel            978   7
mehmet arslan            88
mustafa kemal           482   3   7   9
ali kemal               314   2   4   5   7   8   9
mustafa kemal           482   3   7   9
mustafa kemal           482   3   7   9
mehmet arslan            88
kemal ahmet             506   0   2   3   4   5   9
ali kaan                428   3   9
kemal kaan              740   2   3   4   5   7   8
```