Data Structures and Algorithms (CS/IS F211)

Lab: 8 **Date: 25-03-2014**

Instructions:

- You are expected to use **C/C++ language** only. You will need only g++ , gedit and VI editor for today's lab, if these softwares are not there on your system, you should call the evaluators.
- Your code will be evaluated by the server. So you strictly need to **follow the given Input/Output format**. Any request for recheck on this basis will not be entertained later.
- You will not be allowed to upload solution after 11.00am.
- Upload your code in the format **<full_id_no>.c** (one file per Question) on the under mentioned server

  http://10.1.5.104/domjudge/team

- Login detail
- **Username** : <Full ID No in UPPERCASE>
- **Password**: Will be given to you.

Question 1: HASHING (OPEN ADDRESSING WITH CHAINING)

Consider a student database, where each student record contains **student name and their total mark** obtained in a particular subject. It is decided that the **whole database will be divided into different groups** and the **topper will be found out from the respective groups** who have highest mark in their group. For this, name (unique) of the students are decided as the key. Implement this using **hashing** .To handle **collision** (if any), use **open addressing with chaining.** Under this scheme, if **two names collide** (say `Y' collides with the existing name 'X'), then 'Y' should belongs to the group of 'X' and 'X' should contain address of 'Y'. Similarly if `Z' collide with either `X' or `Y', it will be stored at the end of the list(`X '→`Y' →`Z') and so on**. If 'Y' (new element) does not collide** with any element, then it will form a new group, with 'Y' as the first element of the group. When the table is full then you have to rehash the elements to the new hash table, double the size of old hash table. **Students in a group should not be changed** while rehashing.

(*Note*:
1. Use the hash function ((h'(x)) as described below:-
The weightage for character `A' or `a' as 1, `B' or `b' as 2 and so on. Name should consisting of alphabets (A | a – Z | z) only.
The hash function **h'(x) = hash (key) = (int(key))modulus m**, where `m' is number of slots in the hash table. Example: hash (Amar) = (1+13+1+18) modulus m.

2. Use linear probing mechanism for open addressing **(h(x) = ((h'(x) + i) mod m)**
where i=0,1,2……,m and m is the number of slots.

INPUT FORMAT

<M>
<N>
<Name of student 1>
<Marks of student 1>
<Name of student 2>
<Marks of student 2>
.
.
<Name of student N>
<Marks of student N>

M-Number of slots
N- Number of students

OUTPUT FORMAT

<G> **G- Total Groups**

<Group Number 1> <Name of Topper> <Marks>

<Group Number 2> <Name of Topper> <Marks>

.

.

<Group Number G> <Name of Topper> <Marks>

INPUT.in

2
4
Shikha
82
Sahil
69
Ravi
92
Nikita
88

OUTPUT.out

2
0 Ravi 92
1 Sahil 49

Only for Reference, Not needed in output

Below shows one way of implementation for understanding, there can be other ways of implementing the required hashing method.

if head=1 indicates head of each group

Hash table after inserting Shikha and Sahil as they will not collide, both will be in different groups

Index	Name	Marks	head	next_group_member
0	Shikha	82	head=1	next_group_member=-1
1	Sahil	69	head=1	next_group_member=-1

for Ravi rehash....

After rehashing and inserting other names, Ravi and Nikita collide with Shikha, so they are in same group

0	Shikha	82	head=1	next_group_member=2
1	Sahil	69	head=1	next_group_member=-1
2	Ravi	92	head=0	next_group_member=3
3	Nikita	88	head=0	next_group_member=-1