**Lab 3 – Perfect Hashing**

Please write a program to hash 9-digit social security numbers into 3-digit employee ids without any collision. A collision occurs if two or more different social security numbers are hashed into the same employee id. You need to keep modifying your hashing algorithm until no collision occurs for all those social security numbers in the table. You must use the following table of 12 social security numbers for this programming exercise. After you discover a perfect hashing algorithm, you should print the hash table in two columns with the first column being the 9-digit social security number and the second column the 3-digit employee ids. Your hashing algorithm is perfect only if all 3-digit employee ids are unique.

123456789

234567890

345678912

456789012

567896789

123456798

987654321

887766554

012345678

113456745

665544332

723456789

Hashing is the transformation of a string of characters into a usually shorter fixed-length value or key that represents the original string. Hashing is used to index and retrieve items in a database because it is faster to find the item using the shorter hashed key than to find it using the original value.

A hashing algorithm is considered perfect if no collision occurs for all those input strings. It’s possible to find a perfect hashing algorithm for a small size of input strings such as 12.

How to find a perfect hashing algorithm? For 9-digit numbers, you first try to pick 3 digits out of 9. For example, you may pick first digit, fifth digit, and eighth digit. Then, you may change the picks if you see collisions. You may pick 3 digits and then add a number (11 for example) to it. Please continue changing your algorithm until the hashing is perfect (i.e. without any collision).

==========================================================================.

**How to submit your Lab or Project Assignment (PA)?**

(1) Each program must be well-documented with block comments and proper line comments. The

beginning of each program must have a block comment to show your name, date, and purpose.

The following is an example of block and line comments.

/\*

Author: Dr. Simon Lin

Date: 2/17/2016

Purpose: To perfectly hash 12 social security numbers into 3-digit employee ids with no collision

\*/

int ssn = 0, eid = 0 ; // ssn is Social Security Number, eid is Employee ID

(2) You must submit the following items as attachments through sakai.apu.edu.

(a) All source programs (i.e., all **.java** files), and

(b) One WORD document (i.e., **.doc** or **.docx** file) containing all source programs’ listing and the

output of your hash table. ==========================================================================.

Grading Rubric:

You got \_\_\_ points out of 100 for **CS250 Lab 2**. Thank you for your excellent/good work.

[ ] -10 points for each day late.

[ ] 30 points – Your program must be fully tested until the hashing is perfect without any collision. You must show your hash table in your Word doc.

[ ] 10 points – Your program must be well-documented.

[ ] 30 points – You must follow the program specification to develop your programs properly and completely.

[ ] 20 points – You must submit your WORD document.

[ ] -5 points if your program did not have block comment to show your name, date, and purpose.

[ ] 5 points – Your WORD document must show “**CS250 Lab 2**” and your full name on the **header**.

[ ] 5 points – Your WORD document must be page-numbered on the footer with format “Page 2 of 15” (for example).

==================================================================================.

Please use this Word document as a template for your Word document to be submitted. Please delete everything above.

Please copy all your source programs into here:

/\*

Author: JIE GAO

Date: 2/24/2016

Purpose: To find out a perfect hashing method.

\*/

**public** **class** PerfectHashing {

**public** **static** **void** main(String[] args){

String[] SSN = **new** String[] {"123456789", "234567890", "345678912",

"456789012", "567896789", "123456798", "987654321",

"887766554", "012345678", "113456745", "665544332", "723456789"};

**int**[] EID = **new** **int**[12];

**for**(**int** i = 0; i < 12; i++){

EID[i] = *hashing*(Integer.*parseInt*(SSN[i]));

System.***out***.print("The "+(i+1)+"th SSN is:"+SSN[i]+" ");

System.***out***.println("The "+(i+1)+"th EID is:"+EID[i]);

}

}

**public** **static** **int** hashing(**int** ssn){

**int** eid;

**int** fir, sec, thr;

**int** a,b,c;

fir = ssn/1000000\*13;

sec = (ssn/1000)%1000\*37;

thr = ssn%1000\*83;

eid = 93+(fir+sec+thr)%907;

**return** eid;

}

}

Please copy your perfect hash table output into here:

The 1th SSN is:123456789 The 1th EID is:607

The 2th SSN is:234567890 The 2th EID is:935

The 3th SSN is:345678912 The 3th EID is:148

The 4th SSN is:456789012 The 4th EID is:837

The 5th SSN is:567896789 The 5th EID is:891

The 6th SSN is:123456798 The 6th EID is:447

The 7th SSN is:987654321 The 7th EID is:275

The 8th SSN is:887766554 The 8th EID is:690

The 9th SSN is:012345678 The 9th EID is:356

The 10th SSN is:113456745 The 10th EID is:453

The 11th SSN is:665544332 The 11th EID is:188

The 12th SSN is:723456789 The 12th EID is:244

Please describe your perfect hashing algorithm here:

I took the first 3 digits multiply 13, the second 3 digits multiply 37, and the last 3 digits multiply 83. Then add them together. Then mod by 907, and plus 93.