Team members:
Cong Le Cle155@csu.fullerton.edu
Jack Nguyen nhannguyen7@csu.fullerton.edu
CPSC335-03: Algorithm Engineering
Professor Kevin Wortman
Due date: Friday, November 30, 1 pm.

Project 3- Hashing

Abstract
Throughout this project, we will design, implement, and analyze one algorithm for the hashing problem. The algorithm is called Cuckoo Hashing, presented in class. For the problem, we will design and implement one algorithm in C++, test it on various inputs and complete a hash table with a given input. No algorithm analysis is needed for this project

Part I: Description of the Cuckoo Hashing Algorithm:

There are several versions of cuckoo hashing. The version we learned in class is the simplest, where there are two hash functions, and thus only two places where any given item could be stored in the table.
Let us consider the set of keys to be printable ASCII strings of length no more than 80.
Let us consider the hash table size to be 17.
If key is the string representing the key, then let *keysize* be the *size* of the string and *key[i]* be the ASCII code of the $(i+1)^{th}$ character in the string key read from left to right:

$$key = key_0 key_1 \dots key_{keysize-1}$$

Java using the following hash function on strings

$$val = 31^{keysize-1} \cdot key_0 + 31^{keysize-2} \cdot key_1 + \cdots + 31^1 \cdot key_{keysize-2} + key_{keysize-1}$$

Let us consider two hash functions, f_1 and f_2. Function f_2 will compute the hash value using Java's hash function formula, while the function f_1 computes a different hash value using a different hash function. Function f_1 computes first a large number then it brings the result into the proper range using the formulas below:

$$val = \sum_{i=0}^{keysize-1} key[i] \cdot 31^i$$

$$f_1 = val\%\ tablesize, if\ f_1 < 0\ then\ f_1 = f_1 + tablesize$$

Function f_2 computes first a large number then it brings the result into the proper range using the formulas below:

$$val = \sum_{i=0}^{keysize-1} key[keysize - i - 1] \cdot 31^i$$

$$f_2 = val\%\ tablesize, if\ f_2 < 0\ then\ f_2 = f_2 + tablesize$$

Both functions f_1 and f_2 compute first a large number then it brings the result into the proper range 0..tablesize-1. But we bring the intermediate results into the proper range after each calculation, we do not need to wait until we compute the final result. Also, we can bring the power term 31^{index} into the proper range before multiplying it with key_{index}

Part II: Hashing Tables:

The following strings (given in the input file in6.txt) will be inserted into the hash tables using f_1 for the first table and f_2 for the second table:

Algorithm Engineering
California State University
Fullerton
College of Engineering
and Computer Science
Department of Computer
Science
Dynamic Programming
Monge Properties
String Matching
Matrix Searching
Optimal Tree Construction
Online algorithms
emphasis on
Server Problem
Some related problem
Self-Stabilization
One of the greatest
mysteries in science
Quantum Nature of Universe
In physics and
are known
Cuckoo hashing is fun

	Table T1 t[0]	Table T2 t[1]
[0]	Online algorithms	
[1]		Some related problem
[2]	Self-Stabilization	Monge Properties
[3]	are known	Fullerton
[4]	Quantum Nature of Universe	Server Problem
[5]	In physics and	College of Engineering
[6]	One of the greatest	Optimal Tree Construction
[7]		
[8]		
[9]	Cuckoo hashing is fun	
[10]		
[11]	Algorithm Engineering	Matrix Searching
[12]	Science	
[13]		and Computer Science
[14]	Department of Computer	Dynamic Programming
[15]	emphasis on	mysteries in science
[16]	String Matching	California State University