THOMAS NGO
VAN DO
Project 4

1. Pseudocode:
Cuckoo hashing algorithm
//INPUT: an input file containing strings of characters, one string per line
//OUTPUT: a detailed list of where the strings are inserted

take the input from file and put into table 1
if there is a collision, move the previous string input to table 2. Keep doing it until no more input is remained
if the collision happens again on table 2, recalculate the hash key value and continue doing that until we find the empty spot.

```
bool place_in_hash_tables (char *s)

place ←false                    //return true of the input is placed successfully into table
pos                             //index of the table
index                           //indicating the table number
counter←0                       //using to detect loops
tablesize ←17                   //the size of each table
temp_s                          //string of input
temp                            //temporary variable to hold string for swap

index←0                         //start with table 1
place ←false
pos ←f (temp_s, index)          //calculate the key using hashing function of table 1

while place = true and counter < 2*tablesize
        if t[pos][index] = 0 then
                print string at position “pos” in table 1
                place ←true
                return place
        else do
                print string will be placed at new position “pos” on table 2
                swap the strings between t[pos][index] and temp_s
                index ←1
                pos ←f (temp_s, index)          //find new hash key in table 2

                if t[pos][index] = 0 then
                        place ←true
                        copy string in t[pos][index] to temp_s
                        return placed
                else
                        swap the strings between temp and t[pos][index] and temp_s
                        index ←0 to get back to table 1
                        pos ←f(temp_s, index)           //recalculate the new hash key in table 1
                increment counter
return placed
```

```
size_t f(char *s, size_t index)
//compute the hash functions
//s is the string (key) to which we apply the hash function
//index indicates which hash function will be used
//index = 0 means first hash function
//index = 1 means second hash functions

len                                     //length of string
val                                     //hash value
temp                                    //temporary value to hold information
po                                      //power of number 31
tablesize                               //size of each table

if it is table 1 (index = 0) then
        val ←s[0]
        val ←val mod tablesize
        if val < 0 then
                val ←val + tablesize
        if len = 1
                return val
        for i←1 to i<len do
                temp ←s[i]
                po ←po * 31
                po = po mod tablesize
                if po<0
                        po ←po + tablesize
                val ←val + temp * po
                val ←val mod tablesize
                if val < 0
                        val ←val + tablesize
        return val
else
        calculate the same thing for second hash functions
        only need to change the index of string s to l-i-1
```

the table output from the file in6.txt

	Table 1	Table 2
[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