323.35 Project 3: Radix Sort for String Java

Student: Matthew Flammia

Project Due date: 9/17/2020

****
IV. Main(…)
****

```
Step 0: inFile  open the input file
outFile1  open outFile1 // for the output of sorted data
outFile2  open outFile2 // for observations
use constructor to create two hash tables of LLQueue where each hashTable[i][j] linked
list queue with
a dummy node, and let the head and tail point to the dummy node.
Step 1: firstReading (inFile) // see algorithm below
Step 2: close inFile
Step 3: inFile  open the input file // open the file second time
step 4: S  loadStack (inFile) // see algorithm below
Step 5: printStack (S, outFile2) // Print caption!!! Say what you are printing
Step 6: currentPosition  longestStringLength -1 // Sort from right to left of the
paddedData.
currentTable  0
Step 7: moveStack (currentPosition, currentTable) // move all nodes on the stack to
// the first hash table. See the algorithm below
Step 8: - currentPosition - -
- nextTable  mod (currentTable + 1, 2)
- currentQueue  0
Step 9: // moving nodes from currentTable to nextTable, process queues in the current
table sequentially.
node <-- deleteHead (hashTable[currentTable][currentQueue])
chr <-- getChar (node, currentPosition) // i.e., the character at the currentPosition of
node's data
hashIndex <-- (int) chr or atoi (chr) // cast chr from asci to integer
addTail (hashTable[nextTable][hashIndex], node) //
// add the node at the tail of the queue at hashTable[nextTable][hashIndex]
Step 10: repeat steps 9 until the currentQueue is empty // finish moving all node in
currentQueue.
Step 11: currentQueue ++ // process the next queue in the current hashTable
Step 12: repeat step 9 to step 11 while currentQueue < tableSize
// finish moving all queues from the current table.
Step 13: printTable((hashTable[nextTable], outFile2) // to outFile2
Step 14: currentTable  nextTable
Step 15: repeat step 8 to step 14 while currentPosition >= 0
Step 16: PrintSortedData (hashTable[nextTable], outFile1)
Step 17: close all files
```

Source Code

```
import java.util.NoSuchElementException;
import java.util.Scanner;
import java.io.*;
import java.io.PrintStream;

/**
* Matthew Flammia, 23661371
* CSCI 323.35 Project 3
* To execute, have the source data file in working directory.
* args[0] is the source data file
* args[1] is the final output file name
* args[2] is the observations file name
* Example run command:
* java -cp bin FlammiaM_Project1_Java RadixSortString_Data1.txt outFile1.txt
outFile2.txt
**/

public class FlammiaM_Project3_Java{
      public static void main(String[] args) throws Exception{
            //file setup step 0
            File data = new File(args[0]);
            //hashtable creation
            hashTable sorter = new hashTable();
            //creation of stack step 1
            sorter.firstReading(data);
            LLStack inputData = sorter.loadStack(data);

            //step 5
            inputData.printStack(args[2]);
            //first input of data step 6, 7
            sorter.currentPosition = sorter.longestStringLength - 1;
            sorter.moveStack(inputData, sorter.currentPosition,
sorter.currentTable);

            //updating table information step 8
            sorter.currentPosition = sorter.currentPosition - 1;
            //moving from table 0 to 1 step 9
            while(sorter.currentPosition >= 0){
                  for(int i=0;i<256;i++){

if(!sorter.hashTable[sorter.currentTable][i].isEmpty()){

sorter.moveNextTable(sorter.hashTable[sorter.currentTable][i],sorter.currentPo
sition,sorter.nextTable);
                        }
                  }
                  sorter.currentTable = sorter.tableIndex();
                  sorter.nextTable = sorter.tableIndex();
                  sorter.printTable(args[2]);
                  sorter.currentPosition = sorter.currentPosition - 1;
            }
            //PRINT SORTED DATA
            sorter.printSortedData(args[1]);
      }
}

class listNode{
      //variables
      String data;
      listNode next;
      //constructors
      listNode(String data){
            this.data = data;
      }
      //methods
      void printNode() throws NullPointerException{
            try{
                  System.out.print("("+this.data+", "+this.next.data+")->");
            }
            catch (Exception e){
                  System.out.print("("+this.data+", NULL)->");
            }
      }
}

class LLStack{
      //variables
      listNode top;
      //constructors
      LLStack(){
            this.top = new listNode("dummyNode");
            this.top.next = null;
      }
      //methods
      void push(listNode node){
            node.next = this.top;
            this.top = node;
      }
      listNode pop(){
            //checks if empty; then returns null if is
            if(isEmpty()){
                  return null;
            }
            //makes a temp node, stores the value of top, then sets new top to
top.next
            listNode temp;
            temp = this.top;
            this.top = this.top.next;
            return temp;
      }
      boolean isEmpty(){
            //checks if stack is empty
            if(this.top.next==null){
                  return true;
            }
            return false;
      }
      void printStack(String outFile2) throws FileNotFoundException{
            PrintStream fileOut = new PrintStream(new
FileOutputStream(outFile2,true));
            System.setOut(fileOut);
            System.out.println("Printing Stack...");
            listNode readHead = this.top;
            while(readHead.next != null){
                  readHead.printNode();
                  readHead = readHead.next;
            }
            System.out.print("\n");
            fileOut.close();
      }
}

class LLQueue{
      //variables
      listNode head;
      listNode tail;
      listNode dummy;
      //methods
      LLQueue(){
            //constructor
            dummy = new listNode("dummyNode");
            head = new listNode("head");
            tail = new listNode("tail");
            head.next = dummy;
            tail.next = dummy;
      }
      void insertQ(listNode node){
            //special case for first node insert
            if(this.isEmpty()){
                  dummy.next = node;
                  tail.next = node;
            }
            //case for all other nodes after first
            else{
                  tail.next.next = node;
                  tail.next = node;
            }
      }
      listNode deleteQ(){
            //prevents deleting empty queue
            if(isEmpty()){
                  return null;
            }
            //special case for single node in queue
            if(dummy.next == tail.next){
                  listNode temp = dummy.next;
                  tail.next = dummy;
                  dummy.next = null;
                  return temp;
            }
            //generic node removal
            else{
                  listNode temp = dummy.next;
                  dummy.next = dummy.next.next;
                  return temp;
            }
      }
      boolean isEmpty(){
            if(tail.next == dummy){
                  return true;
            }
            return false;
      }
      void printQueue(int table, int index, String outFile2) throws
FileNotFoundException{
            System.out.print("Table["+table+"]["+index+"]: ");
            listNode readHead = this.head.next;
            while(readHead.next!=this.tail.next){
                  readHead.printNode();
                  readHead = readHead.next;
            }
            readHead.printNode();
            readHead.next.printNode();
            System.out.print("NULL\n");
        }
}
class hashTable{
        LLQueue[][] hashTable;
        int currentTable;
        int nextTable;
        int longestStringLength;
        int currentPosition;
        //constructor
        hashTable(){
            this.hashTable = new LLQueue[2][256];
            for(int i=0;i<2;i++){
                for(int j=0;j<256;j++){
                    this.hashTable[i][j] = new LLQueue();
                }
            }
            this.currentTable = 0;
            this.nextTable = 1;
            this.longestStringLength = 0;
            this.currentPosition = 0;
        }

        //methods
        void firstReading(File inputData) throws FileNotFoundException{
            this.longestStringLength = 0;
            Scanner reader = new Scanner(inputData);
            while(reader.hasNext()){
                String temp = reader.next();
                if(this.longestStringLength < temp.length()){
                    this.longestStringLength = temp.length();
                }
            }
        }
        LLStack loadStack(File inputData) throws FileNotFoundException{
            LLStack stack = new LLStack();
            Scanner reader = new Scanner(inputData);
            while(reader.hasNext()){
                String temp = padString(reader.next());
                stack.push(new listNode(temp));
            }
            return stack;
        }
        void moveStack(LLStack stack, int currentPosition, int currentTable){
            while(!stack.isEmpty()){
                listNode current = stack.pop();
```

```
                  char currentCharacter = getChar(current, currentPosition);
                  int hashIndex = currentCharacter;
                  this.hashTable[currentTable][hashIndex].insertQ(current);
            }
      }
      void moveNextTable(LLQueue queue, int currentPosition, int nextTable){
            while(!queue.isEmpty()){
                  listNode current = queue.deleteQ();
                  current.next = null;
                  char currentCharacter = getChar(current, currentPosition);
                  int hashIndex = currentCharacter;
                  this.hashTable[nextTable][hashIndex].insertQ(current);
            }
      }
      int tableIndex(){
            int table = (this.currentTable + 1) % 2;
            return table;
      }
      char getChar(listNode input, int currentPosition){
            String temp = input.data;
            char current = temp.charAt(currentPosition);
            return current;
      }
      String padString(String data){
            while(data.length()!=this.longestStringLength){
                  data = data + " ";
            }
            return data;
      }
      void printTable(String outFile2) throws FileNotFoundException{
            PrintStream fileOut = new PrintStream(new
FileOutputStream(outFile2,true));
            System.setOut(fileOut);
            System.out.println("Printing Table...");
            for(int i=0;i<256;i++){
                  if(!this.hashTable[this.currentTable][i].isEmpty()){

this.hashTable[this.currentTable][i].printQueue(this.currentTable,i,"outFile1.
txt");
                  }
            }
            fileOut.close();
      }
      void printSortedData(String outFile1) throws FileNotFoundException{
            PrintStream fileOut = new PrintStream(new
FileOutputStream(outFile1,true));
            System.setOut(fileOut);
            for(int i=0;i<256;i++){
                    if(!this.hashTable[this.currentTable][i].isEmpty()){

while(!this.hashTable[this.currentTable][i].isEmpty()){
                              listNode temp =
this.hashTable[this.currentTable][i].deleteQ();
                              System.out.println(temp.data);
                         }
                    }
               }
               fileOut.close();
          }
}
```

Output

```
outFile1 (random letters)
AAbb
AbCdEfG
Acc
BbAa
Bdd
CcaabB
Hijk
XyZzz
ZZZZ
aA
acc
bggff
cCaAbb
jIJkL
xyyk
zxcccc

outFile2 (random letters)
Printing Stack...
(bggff  , Bdd    )->(Bdd    , acc    )->(acc    , Acc    )->(Acc    , jIJkL  )->(jIJkL  , Hijk
)->(Hijk   , ZZZZ   )->(ZZZZ   , xyyk   )->(xyyk   , XyZzz  )->(XyZzz  , zxcccc )->(zxcccc ,
AAbb   )->(AAbb   , aA     )->(aA     , BbAa   )->(BbAa   , CcaabB )->(CcaabB , cCaAbb
)->(cCaAbb , AbCdEfG)->(AbCdEfG, dummyNode)->
Printing Table...
```

```
Table[1][32]: (dummyNode, bggff  )->(bggff  , Bdd    )->(Bdd    , acc    )->(acc    , Acc
)->(Acc    , jIJkL  )->(jIJkL  , Hijk   )->(Hijk   , ZZZZ   )->(ZZZZ   , xyyk   )->(xyyk   , XyZzz
)->(XyZzz  , AAbb   )->(AAbb   , aA     )->(aA     , BbAa   )->(BbAa   , NULL)->NULL
Table[1][66]: (dummyNode, CcaabB )->(CcaabB , NULL)->NULL
Table[1][98]: (dummyNode, cCaAbb )->(cCaAbb , NULL)->NULL
Table[1][99]: (dummyNode, zxcccc )->(zxcccc , NULL)->NULL
Table[1][102]: (dummyNode, AbCdEfG)->(AbCdEfG, NULL)->NULL
Printing Table...
Table[0][32]: (dummyNode, Bdd    )->(Bdd    , acc    )->(acc    , Acc    )->(Acc    , Hijk
)->(Hijk   , ZZZZ   )->(ZZZZ   , xyyk   )->(xyyk   , AAbb   )->(AAbb   , aA     )->(aA     ,
BbAa   )->(BbAa   , NULL)->NULL
Table[0][69]: (dummyNode, AbCdEfG)->(AbCdEfG, NULL)->NULL
Table[0][76]: (dummyNode, jIJkL  )->(jIJkL  , NULL)->NULL
Table[0][98]: (dummyNode, CcaabB )->(CcaabB , cCaAbb )->(cCaAbb , NULL)->NULL
Table[0][99]: (dummyNode, zxcccc )->(zxcccc , NULL)->NULL
Table[0][102]: (dummyNode, bggff  )->(bggff  , NULL)->NULL
Table[0][122]: (dummyNode, XyZzz  )->(XyZzz  , NULL)->NULL
Printing Table...
Table[1][32]: (dummyNode, Bdd    )->(Bdd    , acc    )->(acc    , Acc    )->(Acc    , aA
)->(aA     , NULL)->NULL
Table[1][65]: (dummyNode, cCaAbb )->(cCaAbb , NULL)->NULL
Table[1][90]: (dummyNode, ZZZZ   )->(ZZZZ   , NULL)->NULL
Table[1][97]: (dummyNode, BbAa   )->(BbAa   , CcaabB )->(CcaabB , NULL)->NULL
Table[1][98]: (dummyNode, AAbb   )->(AAbb   , NULL)->NULL
Table[1][99]: (dummyNode, zxcccc )->(zxcccc , NULL)->NULL
Table[1][100]: (dummyNode, AbCdEfG)->(AbCdEfG, NULL)->NULL
Table[1][102]: (dummyNode, bggff  )->(bggff  , NULL)->NULL
Table[1][107]: (dummyNode, Hijk   )->(Hijk   , xyyk   )->(xyyk   , jIJkL  )->(jIJkL  ,
NULL)->NULL
Table[1][122]: (dummyNode, XyZzz  )->(XyZzz  , NULL)->NULL
Printing Table...
Table[0][32]: (dummyNode, aA     )->(aA     , NULL)->NULL
Table[0][65]: (dummyNode, BbAa   )->(BbAa   , NULL)->NULL
Table[0][67]: (dummyNode, AbCdEfG)->(AbCdEfG, NULL)->NULL
Table[0][74]: (dummyNode, jIJkL  )->(jIJkL  , NULL)->NULL
Table[0][90]: (dummyNode, ZZZZ   )->(ZZZZ   , XyZzz  )->(XyZzz  , NULL)->NULL
Table[0][97]: (dummyNode, cCaAbb )->(cCaAbb , CcaabB )->(CcaabB , NULL)->NULL
Table[0][98]: (dummyNode, AAbb   )->(AAbb   , NULL)->NULL
```

```
Table[0][99]: (dummyNode, acc    )->(acc    , Acc    )->(Acc    , zxcccc )->(zxcccc ,
NULL)->NULL
Table[0][100]: (dummyNode, Bdd    )->(Bdd    , NULL)->NULL
Table[0][103]: (dummyNode, bggff  )->(bggff  , NULL)->NULL
Table[0][106]: (dummyNode, Hijk   )->(Hijk   , NULL)->NULL
Table[0][121]: (dummyNode, xyyk   )->(xyyk   , NULL)->NULL
Printing Table...
Table[1][65]: (dummyNode, aA     )->(aA     , AAbb   )->(AAbb   , NULL)->NULL
Table[1][67]: (dummyNode, cCaAbb )->(cCaAbb , NULL)->NULL
Table[1][73]: (dummyNode, jIJkL  )->(jIJkL  , NULL)->NULL
Table[1][90]: (dummyNode, ZZZZ   )->(ZZZZ   , NULL)->NULL
Table[1][98]: (dummyNode, BbAa   )->(BbAa   , AbCdEfG)->(AbCdEfG, NULL)->NULL
Table[1][99]: (dummyNode, CcaabB )->(CcaabB , acc    )->(acc    , Acc    )->(Acc    ,
NULL)->NULL
Table[1][100]: (dummyNode, Bdd    )->(Bdd    , NULL)->NULL
Table[1][103]: (dummyNode, bggff  )->(bggff  , NULL)->NULL
Table[1][105]: (dummyNode, Hijk   )->(Hijk   , NULL)->NULL
Table[1][120]: (dummyNode, zxcccc )->(zxcccc , NULL)->NULL
Table[1][121]: (dummyNode, XyZzz  )->(XyZzz  , xyyk   )->(xyyk   , NULL)->NULL
Printing Table...
Table[0][65]: (dummyNode, AAbb   )->(AAbb   , AbCdEfG)->(AbCdEfG, Acc    )->(Acc
, NULL)->NULL
Table[0][66]: (dummyNode, BbAa   )->(BbAa   , Bdd    )->(Bdd    , NULL)->NULL
Table[0][67]: (dummyNode, CcaabB )->(CcaabB , NULL)->NULL
Table[0][72]: (dummyNode, Hijk   )->(Hijk   , NULL)->NULL
Table[0][88]: (dummyNode, XyZzz  )->(XyZzz  , NULL)->NULL
Table[0][90]: (dummyNode, ZZZZ   )->(ZZZZ   , NULL)->NULL
Table[0][97]: (dummyNode, aA     )->(aA     , acc    )->(acc    , NULL)->NULL
Table[0][98]: (dummyNode, bggff  )->(bggff  , NULL)->NULL
Table[0][99]: (dummyNode, cCaAbb )->(cCaAbb , NULL)->NULL
Table[0][106]: (dummyNode, jIJkL  )->(jIJkL  , NULL)->NULL
Table[0][120]: (dummyNode, xyyk   )->(xyyk   , NULL)->NULL
Table[0][122]: (dummyNode, zxcccc )->(zxcccc , NULL)->NULL
```

outFile1 (names)

```
Aaron
Abeesh
Alexis
Amreen
```

Angel
Archimed
Asher
Bijaya
Bret
Calvin
Carlos
Christopher
Christos
Edwin
Eunhee
Frederick
Gurnoor
Hammad
Hyungbin
Jamil
Jason
Jason
Jephter
Jianhui
Jiawei
Jonathan
Jordon
Joshua
Juan
Justin
Kenley
Kevin
Krzysztof
Lei
Lei
Martin
Matthew
Matthew
Michael
Murgray
Nahian
Naiem
Nectario

```
Rajendra
Rashad
Robert
Roberto
Rupert
Russell
Ryan
Shadman
Shahan
Shelley
Steven
Talha
Tenzin
Thomas
Umair
Vincenzo
Weiting
Wilber
Wong
Ye
Yuhang
Yulin
Zaynab

outFile2
Printing Stack...
(Eunhee     , Krzysztof  )->(Krzysztof  , Gurnoor    )->(Gurnoor    , Shadman
)->(Shadman    , Talha      )->(Talha      , Edwin      )->(Edwin      , Angel      )->(Angel
, Shahan     )->(Shahan     , Jason      )->(Jason      , Alexis     )->(Alexis     , Ryan
)->(Ryan       , Jephter    )->(Jephter    , Vincenzo   )->(Vincenzo   , Shelley    )->(Shelley
, Thomas     )->(Thomas     , Jamil      )->(Jamil      , Amreen     )->(Amreen     , Rajendra
)->(Rajendra   , Jordon     )->(Jordon     , Murgray    )->(Murgray    , Lei        )->(Lei        ,
Joshua     )->(Joshua     , Rashad     )->(Rashad     , Umair      )->(Umair      , Abeesh
)->(Abeesh     , Aaron      )->(Aaron      , Tenzin     )->(Tenzin     , Archimed
)->(Archimed   , Wilber     )->(Wilber     , Asher      )->(Asher      , Bijaya     )->(Bijaya     ,
Carlos     )->(Carlos     , Roberto    )->(Roberto    , Christos   )->(Christos   , Jonathan
)->(Jonathan   , Martin     )->(Martin     , Steven     )->(Steven     , Lei        )->(Lei        ,
Nectario   )->(Nectario   , Zaynab     )->(Zaynab     , Matthew    )->(Matthew    , Calvin
)->(Calvin     , Wong       )->(Wong       , Bret       )->(Bret       , Kenley     )->(Kenley     ,
Jason      )->(Jason      , Juan       )->(Juan       , Justin     )->(Justin     , Russell
)->(Russell    , Yulin      )->(Yulin      , Weiting    )->(Weiting    , Rupert     )->(Rupert     ,
Ye         )->(Ye         , Jiawei     )->(Jiawei     , Michael    )->(Michael    ,
Christopher)->(Christopher, Frederick  )->(Frederick  , Naiem      )->(Naiem      ,
Matthew    )->(Matthew    , Kevin      )->(Kevin      , Yuhang     )->(Yuhang     , Nahian
)->(Nahian     , Jianhui    )->(Jianhui    , Hammad     )->(Hammad     , Robert
)->(Robert     , Hyungbin   )->(Hyungbin   , dummyNode)->
Printing Table...
Table[1][32]: (dummyNode, Eunhee     )->(Eunhee     , Krzysztof  )->(Krzysztof  ,
Gurnoor    )->(Gurnoor    , Shadman    )->(Shadman    , Talha      )->(Talha      , Edwin
)->(Edwin      , Angel      )->(Angel      , Shahan     )->(Shahan     , Jason      )->(Jason
, Alexis     )->(Alexis     , Ryan       )->(Ryan       , Jephter    )->(Jephter    , Vincenzo
)->(Vincenzo   , Shelley    )->(Shelley    , Thomas     )->(Thomas     , Jamil      )->(Jamil
, Amreen     )->(Amreen     , Rajendra   )->(Rajendra   , Jordon     )->(Jordon     ,
Murgray    )->(Murgray    , Lei        )->(Lei        , Joshua     )->(Joshua     , Rashad
)->(Rashad     , Umair      )->(Umair      , Abeesh     )->(Abeesh     , Aaron      )->(Aaron
, Tenzin     )->(Tenzin     , Archimed   )->(Archimed   , Wilber     )->(Wilber     , Asher
)->(Asher      , Bijaya     )->(Bijaya     , Carlos     )->(Carlos     , Roberto    )->(Roberto    ,
Christos   )->(Christos   , Jonathan   )->(Jonathan   , Martin     )->(Martin     , Steven
)->(Steven     , Lei        )->(Lei        , Nectario   )->(Nectario   , Zaynab     )->(Zaynab     ,
Matthew    )->(Matthew    , Calvin     )->(Calvin     , Wong       )->(Wong       , Bret
)->(Bret       , Kenley     )->(Kenley     , Jason      )->(Jason      , Juan       )->(Juan       ,
Justin     )->(Justin     , Russell    )->(Russell    , Yulin      )->(Yulin      , Weiting
)->(Weiting    , Rupert     )->(Rupert     , Ye         )->(Ye         , Jiawei     )->(Jiawei     ,
Michael    )->(Michael    , Frederick  )->(Frederick  , Naiem      )->(Naiem      , Matthew
)->(Matthew    , Kevin      )->(Kevin      , Yuhang     )->(Yuhang     , Nahian     )->(Nahian
, Jianhui    )->(Jianhui    , Hammad     )->(Hammad     , Robert     )->(Robert     ,
Hyungbin   )->(Hyungbin   , NULL)->NULL
Table[1][101]: (dummyNode, Christopher)->(Christopher, NULL)->NULL
Printing Table...
Table[0][32]: (dummyNode, Eunhee     )->(Eunhee     , Gurnoor    )->(Gurnoor    ,
Shadman    )->(Shadman    , Talha      )->(Talha      , Edwin      )->(Edwin      , Angel
)->(Angel      , Shahan     )->(Shahan     , Jason      )->(Jason      , Alexis     )->(Alexis     ,
Ryan       )->(Ryan       , Jephter    )->(Jephter    , Vincenzo   )->(Vincenzo   , Shelley
)->(Shelley    , Thomas     )->(Thomas     , Jamil      )->(Jamil      , Amreen     )->(Amreen
, Rajendra   )->(Rajendra   , Jordon     )->(Jordon     , Murgray    )->(Murgray    , Lei
)->(Lei        , Joshua     )->(Joshua     , Rashad     )->(Rashad     , Umair      )->(Umair
, Abeesh     )->(Abeesh     , Aaron      )->(Aaron      , Tenzin     )->(Tenzin     , Archimed
)->(Archimed   , Wilber     )->(Wilber     , Asher      )->(Asher      , Bijaya     )->(Bijaya     ,
Carlos     )->(Carlos     , Roberto    )->(Roberto    , Christos   )->(Christos   , Jonathan
)->(Jonathan   , Martin     )->(Martin     , Steven     )->(Steven     , Lei        )->(Lei        ,
Nectario   )->(Nectario   , Zaynab     )->(Zaynab     , Matthew    )->(Matthew    , Calvin
)->(Calvin     , Wong       )->(Wong       , Bret       )->(Bret       , Kenley     )->(Kenley     ,
Jason      )->(Jason      , Juan       )->(Juan       , Justin     )->(Justin     , Russell
)->(Russell    , Yulin      )->(Yulin      , Weiting    )->(Weiting    , Rupert     )->(Rupert     ,
Ye         )->(Ye         , Jiawei     )->(Jiawei     , Michael    )->(Michael    , Naiem
)->(Naiem      , Matthew    )->(Matthew    , Kevin      )->(Kevin      , Yuhang     )->(Yuhang
, Nahian     )->(Nahian     , Jianhui    )->(Jianhui    , Hammad     )->(Hammad     , Robert
)->(Robert     , Hyungbin   )->(Hyungbin   , NULL)->NULL
Table[0][102]: (dummyNode, Krzysztof  )->(Krzysztof  , NULL)->NULL
Table[0][104]: (dummyNode, Christopher)->(Christopher, NULL)->NULL
Table[0][107]: (dummyNode, Frederick  )->(Frederick  , NULL)->NULL
Printing Table...
Table[1][32]: (dummyNode, Eunhee     )->(Eunhee     , Gurnoor    )->(Gurnoor    ,
Shadman    )->(Shadman    , Talha      )->(Talha      , Edwin      )->(Edwin      , Angel
)->(Angel      , Shahan     )->(Shahan     , Jason      )->(Jason      , Alexis     )->(Alexis     ,
Ryan       )->(Ryan       , Jephter    )->(Jephter    , Shelley    )->(Shelley    , Thomas
)->(Thomas     , Jamil      )->(Jamil      , Amreen     )->(Amreen     , Jordon     )->(Jordon
, Murgray    )->(Murgray    , Lei        )->(Lei        , Joshua     )->(Joshua     , Rashad
)->(Rashad     , Umair      )->(Umair      , Abeesh     )->(Abeesh     , Aaron      )->(Aaron
, Tenzin     )->(Tenzin     , Wilber     )->(Wilber     , Asher      )->(Asher      , Bijaya
)->(Bijaya     , Carlos     )->(Carlos     , Roberto    )->(Roberto    , Martin     )->(Martin     ,
Steven     )->(Steven     , Lei        )->(Lei        , Zaynab     )->(Zaynab     , Matthew
)->(Matthew    , Calvin     )->(Calvin     , Wong       )->(Wong       , Bret       )->(Bret       ,
Kenley     )->(Kenley     , Jason      )->(Jason      , Juan       )->(Juan       , Justin
)->(Justin     , Russell    )->(Russell    , Yulin      )->(Yulin      , Weiting    )->(Weiting    ,
Rupert     )->(Rupert     , Ye         )->(Ye         , Jiawei     )->(Jiawei     , Michael
)->(Michael    , Naiem      )->(Naiem      , Matthew    )->(Matthew    , Kevin      )->(Kevin
, Yuhang     )->(Yuhang     , Nahian     )->(Nahian     , Jianhui    )->(Jianhui    , Hammad
)->(Hammad     , Robert     )->(Robert     , NULL)->NULL
Table[1][97]: (dummyNode, Rajendra   )->(Rajendra   , NULL)->NULL
Table[1][99]: (dummyNode, Frederick  )->(Frederick  , NULL)->NULL
Table[1][100]: (dummyNode, Archimed   )->(Archimed   , NULL)->NULL
Table[1][110]: (dummyNode, Jonathan   )->(Jonathan   , Hyungbin   )->(Hyungbin   ,
NULL)->NULL
Table[1][111]: (dummyNode, Vincenzo   )->(Vincenzo   , Nectario   )->(Nectario   ,
Krzysztof  )->(Krzysztof  , NULL)->NULL
Table[1][112]: (dummyNode, Christopher)->(Christopher, NULL)->NULL
```

```
Table[1][115]: (dummyNode, Christos   )->(Christos   , NULL)->NULL
Printing Table...
Table[0][32]: (dummyNode, Eunhee     )->(Eunhee     , Talha      )->(Talha      , Edwin
)->(Edwin      , Angel      )->(Angel      , Shahan     )->(Shahan     , Jason      )->(Jason
, Alexis     )->(Alexis     , Ryan       )->(Ryan       , Thomas     )->(Thomas     , Jamil
)->(Jamil      , Amreen     )->(Amreen     , Jordon     )->(Jordon     , Lei        )->(Lei        ,
Joshua     )->(Joshua     , Rashad     )->(Rashad     , Umair      )->(Umair      , Abeesh
)->(Abeesh     , Aaron      )->(Aaron      , Tenzin     )->(Tenzin     , Wilber     )->(Wilber     ,
Asher      )->(Asher      , Bijaya     )->(Bijaya     , Carlos     )->(Carlos     , Martin
)->(Martin     , Steven     )->(Steven     , Lei        )->(Lei        , Zaynab     )->(Zaynab     ,
Calvin     )->(Calvin     , Wong       )->(Wong       , Bret       )->(Bret       , Kenley
)->(Kenley     , Jason      )->(Jason      , Juan       )->(Juan       , Justin     )->(Justin     ,
Yulin      )->(Yulin      , Rupert     )->(Rupert     , Ye         )->(Ye         , Jiawei     )->(Jiawei
, Naiem      )->(Naiem      , Kevin      )->(Kevin      , Yuhang     )->(Yuhang     , Nahian
)->(Nahian     , Hammad     )->(Hammad     , Robert     )->(Robert     , NULL)->NULL
Table[0][97]: (dummyNode, Jonathan   )->(Jonathan   , NULL)->NULL
Table[0][101]: (dummyNode, Archimed   )->(Archimed   , NULL)->NULL
Table[0][103]: (dummyNode, Weiting    )->(Weiting    , NULL)->NULL
Table[0][105]: (dummyNode, Jianhui    )->(Jianhui    , Frederick  )->(Frederick  ,
Hyungbin   )->(Hyungbin   , Nectario   )->(Nectario   , NULL)->NULL
Table[0][108]: (dummyNode, Russell    )->(Russell    , Michael    )->(Michael    ,
NULL)->NULL
Table[0][110]: (dummyNode, Shadman    )->(Shadman    , NULL)->NULL
Table[0][111]: (dummyNode, Roberto    )->(Roberto    , Christopher)->(Christopher,
Christos   )->(Christos   , NULL)->NULL
Table[0][114]: (dummyNode, Gurnoor    )->(Gurnoor    , Jephter    )->(Jephter    ,
Rajendra   )->(Rajendra   , NULL)->NULL
Table[0][116]: (dummyNode, Krzysztof  )->(Krzysztof  , NULL)->NULL
Table[0][119]: (dummyNode, Matthew    )->(Matthew    , Matthew    )->(Matthew    ,
NULL)->NULL
Table[0][121]: (dummyNode, Shelley    )->(Shelley    , Murgray    )->(Murgray    ,
NULL)->NULL
Table[0][122]: (dummyNode, Vincenzo   )->(Vincenzo   , NULL)->NULL
Printing Table...
Table[1][32]: (dummyNode, Talha      )->(Talha      , Edwin      )->(Edwin      , Angel
)->(Angel      , Jason      )->(Jason      , Ryan       )->(Ryan       , Jamil      )->(Jamil      ,
Lei        )->(Lei        , Umair      )->(Umair      , Aaron      )->(Aaron      , Asher
)->(Asher      , Lei        )->(Lei        , Wong       )->(Wong       , Bret       )->(Bret       ,
Jason    )->(Jason    , Juan     )->(Juan     , Yulin    )->(Yulin    , Ye       )->(Ye
, Naiem    )->(Naiem    , Kevin    )->(Kevin    , NULL)->NULL
Table[1][97]: (dummyNode, Joshua   )->(Joshua   , Bijaya   )->(Bijaya   , Shadman
)->(Shadman  , Murgray  )->(Murgray  , NULL)->NULL
Table[1][98]: (dummyNode, Zaynab   )->(Zaynab   , Hyungbin )->(Hyungbin ,
NULL)->NULL
Table[1][100]: (dummyNode, Rashad   )->(Rashad   , Hammad   )->(Hammad   ,
Rajendra )->(Rajendra , NULL)->NULL
Table[1][101]: (dummyNode, Eunhee   )->(Eunhee   , Michael  )->(Michael  ,
Jephter  )->(Jephter  , Matthew  )->(Matthew  , Matthew  )->(Matthew  , Shelley
)->(Shelley  , NULL)->NULL
Table[1][103]: (dummyNode, Yuhang   )->(Yuhang   , NULL)->NULL
Table[1][104]: (dummyNode, Abeesh   )->(Abeesh   , Jonathan )->(Jonathan ,
NULL)->NULL
Table[1][105]: (dummyNode, Jiawei   )->(Jiawei   , NULL)->NULL
Table[1][108]: (dummyNode, Russell  )->(Russell  , NULL)->NULL
Table[1][109]: (dummyNode, Archimed )->(Archimed , NULL)->NULL
Table[1][110]: (dummyNode, Shahan   )->(Shahan   , Amreen   )->(Amreen   ,
Jordon   )->(Jordon   , Tenzin   )->(Tenzin   , Martin   )->(Martin   , Steven
)->(Steven   , Calvin   )->(Calvin   , Justin   )->(Justin   , Nahian   )->(Nahian   ,
Weiting  )->(Weiting  , Vincenzo )->(Vincenzo , NULL)->NULL
Table[1][111]: (dummyNode, Gurnoor  )->(Gurnoor  , NULL)->NULL
Table[1][114]: (dummyNode, Wilber   )->(Wilber   , Frederick )->(Frederick , Nectario
)->(Nectario , NULL)->NULL
Table[1][115]: (dummyNode, Alexis   )->(Alexis   , Thomas   )->(Thomas   , Carlos
)->(Carlos   , NULL)->NULL
Table[1][116]: (dummyNode, Rupert   )->(Rupert   , Robert   )->(Robert   , Roberto
)->(Roberto  , Christopher)->(Christopher, Christos )->(Christos , NULL)->NULL
Table[1][117]: (dummyNode, Jianhui  )->(Jianhui  , NULL)->NULL
Table[1][121]: (dummyNode, Kenley   )->(Kenley   , NULL)->NULL
Table[1][122]: (dummyNode, Krzysztof )->(Krzysztof , NULL)->NULL
Printing Table...
Table[0][32]: (dummyNode, Ryan     )->(Ryan     , Lei      )->(Lei      , Lei      )->(Lei
, Wong     )->(Wong     , Bret     )->(Bret     , Juan     )->(Juan     , Ye       )->(Ye
, NULL)->NULL
Table[0][97]: (dummyNode, Talha    )->(Talha    , Zaynab   )->(Zaynab   , Rashad
)->(Rashad   , Hammad   )->(Hammad   , Michael  )->(Michael  , Shahan
)->(Shahan   , Nahian   )->(Nahian   , Nectario )->(Nectario , Thomas
)->(Thomas   , NULL)->NULL
```

```
Table[0][101]: (dummyNode, Eunhee     )->(Eunhee     , Jiawei     )->(Jiawei     , Russell
)->(Russell    , Amreen     )->(Amreen     , Steven     )->(Steven     , Vincenzo
)->(Vincenzo   , Wilber     )->(Wilber     , Frederick  )->(Frederick  , Kenley     )->(Kenley
, NULL)->NULL
Table[0][103]: (dummyNode, Hyungbin   )->(Hyungbin   , NULL)->NULL
Table[0][104]: (dummyNode, Matthew    )->(Matthew    , Matthew    )->(Matthew    ,
Jianhui    )->(Jianhui    , NULL)->NULL
Table[0][105]: (dummyNode, Archimed   )->(Archimed   , Tenzin     )->(Tenzin     , Martin
)->(Martin     , Calvin     )->(Calvin     , Justin     )->(Justin     , Weiting    )->(Weiting    ,
Alexis     )->(Alexis     , NULL)->NULL
Table[0][108]: (dummyNode, Angel      )->(Angel      , Jamil      )->(Jamil      , Shelley
)->(Shelley    , NULL)->NULL
Table[0][109]: (dummyNode, Naiem      )->(Naiem      , Shadman    )->(Shadman    ,
NULL)->NULL
Table[0][110]: (dummyNode, Edwin      )->(Edwin      , Jason      )->(Jason      , Aaron
)->(Aaron      , Jason      )->(Jason      , Yulin      )->(Yulin      , Kevin      )->(Kevin      ,
Rajendra   )->(Rajendra   , Yuhang     )->(Yuhang     , NULL)->NULL
Table[0][111]: (dummyNode, Jordon     )->(Jordon     , Gurnoor    )->(Gurnoor    , Carlos
)->(Carlos     , NULL)->NULL
Table[0][114]: (dummyNode, Umair      )->(Umair      , Asher      )->(Asher      , Murgray
)->(Murgray    , Rupert     )->(Rupert     , Robert     )->(Robert     , Roberto    )->(Roberto
, NULL)->NULL
Table[0][115]: (dummyNode, Abeesh     )->(Abeesh     , Christopher)->(Christopher,
Christos   )->(Christos   , Krzysztof  )->(Krzysztof  , NULL)->NULL
Table[0][116]: (dummyNode, Jephter    )->(Jephter    , Jonathan   )->(Jonathan   ,
NULL)->NULL
Table[0][117]: (dummyNode, Joshua     )->(Joshua     , NULL)->NULL
Table[0][121]: (dummyNode, Bijaya     )->(Bijaya     , NULL)->NULL
Printing Table...
Table[1][32]: (dummyNode, Lei        )->(Lei        , Lei        )->(Lei        , Ye         )->(Ye
, NULL)->NULL
Table[1][97]: (dummyNode, Yuhang     )->(Yuhang     , Jonathan   )->(Jonathan   , Bijaya
)->(Bijaya     , NULL)->NULL
Table[1][98]: (dummyNode, Wilber     )->(Wilber     , NULL)->NULL
Table[1][99]: (dummyNode, Vincenzo   )->(Vincenzo   , NULL)->NULL
Table[1][100]: (dummyNode, Frederick  )->(Frederick  , Shadman    )->(Shadman    ,
Jordon     )->(Jordon     , NULL)->NULL
Table[1][101]: (dummyNode, Amreen     )->(Amreen     , Angel      )->(Angel      , Naiem
)->(Naiem      , Rajendra   )->(Rajendra   , Asher      )->(Asher      , Rupert     )->(Rupert
, Robert     )->(Robert     , Roberto    )->(Roberto    , Abeesh     )->(Abeesh     ,
NULL)->NULL
Table[1][103]: (dummyNode, Wong       )->(Wong       , Murgray    )->(Murgray    ,
NULL)->NULL
Table[1][104]: (dummyNode, Talha      )->(Talha      , Rashad     )->(Rashad     , Michael
)->(Michael    , Shahan     )->(Shahan     , Eunhee     )->(Eunhee     , Archimed
)->(Archimed   , Jephter    )->(Jephter    , Joshua     )->(Joshua     , NULL)->NULL
Table[1][105]: (dummyNode, Nahian     )->(Nahian     , Jamil      )->(Jamil      , Edwin
)->(Edwin      , Yulin      )->(Yulin      , Kevin      )->(Kevin      , Umair      )->(Umair      ,
Christopher)->(Christopher, Christos   )->(Christos   , NULL)->NULL
Table[1][108]: (dummyNode, Kenley     )->(Kenley     , Shelley    )->(Shelley    , Carlos
)->(Carlos     , NULL)->NULL
Table[1][109]: (dummyNode, Hammad     )->(Hammad     , Thomas     )->(Thomas     ,
NULL)->NULL
Table[1][110]: (dummyNode, Ryan       )->(Ryan       , Juan       )->(Juan       , Zaynab
)->(Zaynab     , Hyungbin   )->(Hyungbin   , Jianhui    )->(Jianhui    , Gurnoor
)->(Gurnoor    , NULL)->NULL
Table[1][111]: (dummyNode, Jason      )->(Jason      , Aaron      )->(Aaron      , Jason
)->(Jason      , NULL)->NULL
Table[1][115]: (dummyNode, Russell    )->(Russell    , NULL)->NULL
Table[1][116]: (dummyNode, Bret       )->(Bret       , Nectario   )->(Nectario   , Matthew
)->(Matthew    , Matthew    )->(Matthew    , Martin     )->(Martin     , Justin     )->(Justin
, Weiting    )->(Weiting    , NULL)->NULL
Table[1][118]: (dummyNode, Steven     )->(Steven     , Calvin     )->(Calvin     ,
NULL)->NULL
Table[1][119]: (dummyNode, Jiawei     )->(Jiawei     , NULL)->NULL
Table[1][120]: (dummyNode, Alexis     )->(Alexis     , NULL)->NULL
Table[1][121]: (dummyNode, Krzysztof  )->(Krzysztof  , NULL)->NULL
Table[1][122]: (dummyNode, Tenzin     )->(Tenzin     , NULL)->NULL
Printing Table...
Table[0][32]: (dummyNode, Ye         )->(Ye         , NULL)->NULL
Table[0][97]: (dummyNode, Shadman    )->(Shadman    , Shahan     )->(Shahan     ,
Umair      )->(Umair      , Ryan       )->(Ryan       , Juan       )->(Juan       , Jianhui
)->(Jianhui    , Jiawei     )->(Jiawei     , NULL)->NULL
Table[0][98]: (dummyNode, Robert     )->(Robert     , Roberto    )->(Roberto    ,
NULL)->NULL
Table[0][99]: (dummyNode, Michael    )->(Michael    , Archimed   )->(Archimed   ,
Nectario   )->(Nectario   , NULL)->NULL
```

```
Table[0][101]: (dummyNode, Frederick  )->(Frederick  , Abeesh     )->(Abeesh     ,
Shelley    )->(Shelley    , Bret       )->(Bret       , Steven     )->(Steven     , Alexis
)->(Alexis     , NULL)->NULL
Table[0][103]: (dummyNode, Angel      )->(Angel      , NULL)->NULL
Table[0][104]: (dummyNode, Yuhang     )->(Yuhang     , Asher      )->(Asher      , Nahian
)->(Nahian     , NULL)->NULL
Table[0][105]: (dummyNode, Lei        )->(Lei        , Lei        )->(Lei        , Naiem
)->(Naiem      , Weiting    )->(Weiting    , NULL)->NULL
Table[0][106]: (dummyNode, Bijaya     )->(Bijaya     , Rajendra   )->(Rajendra   ,
NULL)->NULL
Table[0][108]: (dummyNode, Wilber     )->(Wilber     , Talha      )->(Talha      , Yulin
)->(Yulin      , Calvin     )->(Calvin     , NULL)->NULL
Table[0][109]: (dummyNode, Jamil      )->(Jamil      , Hammad     )->(Hammad     ,
NULL)->NULL
Table[0][110]: (dummyNode, Jonathan   )->(Jonathan   , Vincenzo   )->(Vincenzo   ,
Wong       )->(Wong       , Eunhee     )->(Eunhee     , Kenley     )->(Kenley     , Tenzin
)->(Tenzin     , NULL)->NULL
Table[0][111]: (dummyNode, Thomas     )->(Thomas     , NULL)->NULL
Table[0][112]: (dummyNode, Rupert     )->(Rupert     , Jephter    )->(Jephter    ,
NULL)->NULL
Table[0][114]: (dummyNode, Jordon     )->(Jordon     , Amreen     )->(Amreen     ,
Murgray    )->(Murgray    , Christopher)->(Christopher, Christos   )->(Christos   , Carlos
)->(Carlos     , Gurnoor    )->(Gurnoor    , Aaron      )->(Aaron      , Martin     )->(Martin
, NULL)->NULL
Table[0][115]: (dummyNode, Rashad     )->(Rashad     , Joshua     )->(Joshua     , Jason
)->(Jason      , Jason      )->(Jason      , Russell    )->(Russell    , Justin     )->(Justin     ,
NULL)->NULL
Table[0][116]: (dummyNode, Matthew    )->(Matthew    , Matthew    )->(Matthew    ,
NULL)->NULL
Table[0][117]: (dummyNode, Hyungbin   )->(Hyungbin   , NULL)->NULL
Table[0][118]: (dummyNode, Kevin      )->(Kevin      , NULL)->NULL
Table[0][119]: (dummyNode, Edwin      )->(Edwin      , NULL)->NULL
Table[0][121]: (dummyNode, Zaynab     )->(Zaynab     , NULL)->NULL
Table[0][122]: (dummyNode, Krzysztof  )->(Krzysztof  , NULL)->NULL
Printing Table...
Table[1][97]: (dummyNode, Nahian     )->(Nahian     , Naiem      )->(Naiem      , Rajendra
)->(Rajendra   , Talha      )->(Talha      , Calvin     )->(Calvin     , Jamil      )->(Jamil      ,
Hammad     )->(Hammad     , Carlos     )->(Carlos     , Aaron      )->(Aaron      , Martin
)->(Martin     , Rashad     )->(Rashad     , Jason      )->(Jason      , Jason      )->(Jason
, Matthew    )->(Matthew    , Matthew    )->(Matthew    , Zaynab     )->(Zaynab     ,
NULL)->NULL
Table[1][98]: (dummyNode, Abeesh     )->(Abeesh     , NULL)->NULL
Table[1][100]: (dummyNode, Edwin      )->(Edwin      , NULL)->NULL
Table[1][101]: (dummyNode, Ye         )->(Ye         , Nectario   )->(Nectario   , Lei
)->(Lei        , Lei        )->(Lei        , Weiting    )->(Weiting    , Kenley     )->(Kenley     ,
Tenzin     )->(Tenzin     , Jephter    )->(Jephter    , Kevin      )->(Kevin      , NULL)->NULL
Table[1][104]: (dummyNode, Shadman    )->(Shadman    , Shahan     )->(Shahan     ,
Shelley    )->(Shelley    , Thomas     )->(Thomas     , Christopher)->(Christopher,
Christos   )->(Christos   , NULL)->NULL
Table[1][105]: (dummyNode, Jianhui    )->(Jianhui    , Jiawei     )->(Jiawei     , Michael
)->(Michael    , Bijaya     )->(Bijaya     , Wilber     )->(Wilber     , Vincenzo   )->(Vincenzo
, NULL)->NULL
Table[1][108]: (dummyNode, Alexis     )->(Alexis     , NULL)->NULL
Table[1][109]: (dummyNode, Umair      )->(Umair      , Amreen     )->(Amreen     ,
NULL)->NULL
Table[1][110]: (dummyNode, Angel      )->(Angel      , NULL)->NULL
Table[1][111]: (dummyNode, Robert     )->(Robert     , Roberto    )->(Roberto    ,
Jonathan   )->(Jonathan   , Wong       )->(Wong       , Jordon     )->(Jordon     , Joshua
)->(Joshua     , NULL)->NULL
Table[1][114]: (dummyNode, Archimed   )->(Archimed   , Frederick  )->(Frederick  , Bret
)->(Bret       , Krzysztof  )->(Krzysztof  , NULL)->NULL
Table[1][115]: (dummyNode, Asher      )->(Asher      , NULL)->NULL
Table[1][116]: (dummyNode, Steven     )->(Steven     , NULL)->NULL
Table[1][117]: (dummyNode, Juan       )->(Juan       , Yuhang     )->(Yuhang     , Yulin
)->(Yulin      , Eunhee     )->(Eunhee     , Rupert     )->(Rupert     , Murgray    )->(Murgray
, Gurnoor    )->(Gurnoor    , Russell    )->(Russell    , Justin     )->(Justin     ,
NULL)->NULL
Table[1][121]: (dummyNode, Ryan       )->(Ryan       , Hyungbin   )->(Hyungbin   ,
NULL)->NULL
Printing Table...
Table[0][65]: (dummyNode, Aaron      )->(Aaron      , Abeesh     )->(Abeesh     , Alexis
)->(Alexis     , Amreen     )->(Amreen     , Angel      )->(Angel      , Archimed
)->(Archimed   , Asher      )->(Asher      , NULL)->NULL
Table[0][66]: (dummyNode, Bijaya     )->(Bijaya     , Bret       )->(Bret       , NULL)->NULL
Table[0][67]: (dummyNode, Calvin     )->(Calvin     , Carlos     )->(Carlos     ,
Christopher)->(Christopher, Christos   )->(Christos   , NULL)->NULL
Table[0][69]: (dummyNode, Edwin      )->(Edwin      , Eunhee     )->(Eunhee     ,
NULL)->NULL
```

```
Table[0][70]: (dummyNode, Frederick  )->(Frederick  , NULL)->NULL
Table[0][71]: (dummyNode, Gurnoor    )->(Gurnoor    , NULL)->NULL
Table[0][72]: (dummyNode, Hammad     )->(Hammad     , Hyungbin   )->(Hyungbin   ,
NULL)->NULL
Table[0][74]: (dummyNode, Jamil      )->(Jamil      , Jason      )->(Jason      , Jason
)->(Jason      , Jephter    )->(Jephter    , Jianhui    )->(Jianhui    , Jiawei     )->(Jiawei     ,
Jonathan   )->(Jonathan   , Jordon     )->(Jordon     , Joshua     )->(Joshua     , Juan
)->(Juan       , Justin     )->(Justin     , NULL)->NULL
Table[0][75]: (dummyNode, Kenley     )->(Kenley     , Kevin      )->(Kevin      , Krzysztof
)->(Krzysztof  , NULL)->NULL
Table[0][76]: (dummyNode, Lei        )->(Lei        , Lei        )->(Lei        , NULL)->NULL
Table[0][77]: (dummyNode, Martin     )->(Martin     , Matthew    )->(Matthew    , Matthew
)->(Matthew    , Michael    )->(Michael    , Murgray    )->(Murgray    , NULL)->NULL
Table[0][78]: (dummyNode, Nahian     )->(Nahian     , Naiem      )->(Naiem      , Nectario
)->(Nectario   , NULL)->NULL
Table[0][82]: (dummyNode, Rajendra   )->(Rajendra   , Rashad     )->(Rashad     ,
Robert     )->(Robert     , Roberto    )->(Roberto    , Rupert     )->(Rupert     , Russell
)->(Russell    , Ryan       )->(Ryan       , NULL)->NULL
Table[0][83]: (dummyNode, Shadman    )->(Shadman    , Shahan     )->(Shahan     ,
Shelley    )->(Shelley    , Steven     )->(Steven     , NULL)->NULL
Table[0][84]: (dummyNode, Talha      )->(Talha      , Tenzin     )->(Tenzin     , Thomas
)->(Thomas     , NULL)->NULL
Table[0][85]: (dummyNode, Umair      )->(Umair      , NULL)->NULL
Table[0][86]: (dummyNode, Vincenzo   )->(Vincenzo   , NULL)->NULL
Table[0][87]: (dummyNode, Weiting    )->(Weiting    , Wilber     )->(Wilber     , Wong
)->(Wong       , NULL)->NULL
Table[0][89]: (dummyNode, Ye         )->(Ye         , Yuhang     )->(Yuhang     , Yulin
)->(Yulin      , NULL)->NULL
Table[0][90]: (dummyNode, Zaynab     )->(Zaynab     , NULL)->NULL
```