**package** jonahKubath\_A4;

**import** java.io.File;

**import** java.io.FileNotFoundException;

**import** java.util.HashMap;

**import** java.util.NoSuchElementException;

**import** java.util.Scanner;

**public** **class** Main {

//Create a HashMap to hold the start variables

**static** HashMap<String, Node> *map* = **new** HashMap<String, Node>(); //Maps a node the index in the table

**static** HashMap<String, Integer> *values* = **new** HashMap<String, Integer>(); //Maps an input to a column in the table

**static** String *table*[][] = **null**; //Holds the transition rules

**static** String *input* = ""; //Input sentence

**static** String *start* = "q0"; //Starting position

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

System.***out***.println("Running DFA");

String f = "";

**int** fileCount = 2; //Which file to read data from

**if**(fileCount == 0) {

f = "ex\_5.3.3.txt";

}

**else** **if**(fileCount == 1) {

f = "ex\_5.3.4.txt";

}

**else** **if**(fileCount == 2) {

f = "ex\_5.3.5.txt";

}

*readData*(f);

*printData*();

*runDFA*(*input*);

}

/\*\*

\* Iterate through the characters in the input string and evaluate them

\* based on the table

\* **@param** sent The input string

\*/

**public** **static** **void** runDFA(String sent) {

String val = ""; // Individual transition

String cur = *start*; // Current node name

**int** mapIndex = 0; // Row index to the table

**int** valueIndex = 0; // Column index to the table

System.***out***.println("Input String: " + sent);

**while**(sent.length() != 0) {

System.***out***.printf("[%s -> %s]\n", cur, sent); // Print current state

val = sent.charAt(0) + ""; //Get the current value

mapIndex = *map*.get(cur).getIndex(); //Get index of the current node

valueIndex = *values*.get(val); //Get index of the value

cur = *table*[mapIndex][valueIndex]; //Get the new node after the transition

sent = sent.substring(1, sent.length()); //Remove the value used

}

//Print the final state

System.***out***.printf("[%s -> %s]\n", cur, "");

**if**(*map*.get(cur).isFinal()) {

System.***out***.println("ACCEPT");

}

**else** {

System.***out***.println("REJECT");

}

}

/\*\*

\* Read the data from the file

\* Line 0: number of nodes

\* Line 1: Final states

\* Line 2 - n nodes: current node and then nodes evaluated with each input

\* Next Line: input values

\* Final line: input string

\* **@param** filename The filename to open and read

\* **@throws** FileNotFoundException File wasn't found

\*/

**public** **static** **void** readData(String filename) **throws** FileNotFoundException {

File f = **null**;

Scanner scan = **null**;

**try** {

f = **new** File(filename);

scan = **new** Scanner(f);

}

**catch**(FileNotFoundException e) {

System.***out***.println("File: " + filename + " was not found");

System.*exit*(0);

}

**boolean** cont = **true**; // Is there more data to process

**boolean** isFinal = **false**; //Is the node a final state

String line = ""; // Temporary buffer

String split[]; // buffer split by spaces

**int** count = 0; // Count of the current node

**int** total = 0; // Total number of nodes

//Get the number of nodes from the first line

line = scan.nextLine();

total = Integer.*parseInt*(line);

//Read the final states

line = scan.nextLine();

String finalState[] = line.split(" ");

//Read the table

**while**(cont && count < total) {

**try** {

//Get the line

line = scan.nextLine();

//Break it up

split = line.split(" ");

isFinal = *searchFinalStates*(finalState, split[0]);

*map*.put(split[0], **new** Node(count, isFinal));

**if**(*table* == **null**) {

*table* = **new** String[total][split.length - 1];

}

**for**(**int** i = 0; i < split.length - 1; i++) {

*table*[count][i] = split[i+1];

}

count++;

}

**catch**(NoSuchElementException e) {

cont = **false**;

}

}

//Read the input values

line = scan.nextLine();

split = line.split(" ");

**for**(**int** i = 0; i < split.length; i++) {

*values*.put(split[i], i);

}

*input* = scan.nextLine();

*input* = *input*.trim();

scan.close();

}

/\*\*

\* Search the array of final states for the given string

\* **@param** states The array that contains all the final states

\* **@param** name The name of the node to look for

\* **@return** True if the node is a final state, false otherwise

\*/

**public** **static** **boolean** searchFinalStates(String states[], String name) {

**boolean** val = **false**;

//Iterate through the final states

**for**(**int** i = 0; i < states.length; i++) {

**if**(states[i].compareTo(name) == 0) {

val = **true**;

**break**;

}

}

**return** val;

}

/\*\*

\* HashMap iteration taken from:

\* https://stackoverflow.com/questions/1066589/iterate-through-a-hashmap

\*

\* Iterate over the nodes

\* Print: Node name, each transition, and whether or not it is a final state

\*/

**public** **static** **void** printData() {

System.***out***.println("Print Data");

//Print header information

System.***out***.printf("%-5s", "Key");

**for**(String key : *values*.keySet()) {

System.***out***.printf("%-5s", key);

}

System.***out***.println("isFinal");

//Print table

**for** (HashMap.Entry<String, Node> entry : *map*.entrySet()) {

String key = entry.getKey();

Node value = entry.getValue();

System.***out***.printf("%-5s", key);

**for**(**int** i = 0; i < *table*[value.getIndex()].length; i++) {

System.***out***.printf("%-5s", *table*[value.getIndex()][i]);

}

System.***out***.println(value.isFinal());

}

System.***out***.println();

}

}

**package** jonahKubath\_A4;

**public** **class** Node {

**int** index = 0;

**boolean** isFinal = **false**;

**public** Node() {

}

**public** Node(**int** index, **boolean** isFinal) {

**this**.index = index;

**this**.isFinal = isFinal;

}

**public** Node(**int** index) {

**this**.index = index;

isFinal = **false**;

}

**public** **int** getIndex() {

**return** index;

}

**public** **void** setIndex(**int** index) {

**this**.index = index;

}

**public** **boolean** isFinal() {

**return** isFinal;

}

**public** **void** setFinal(**boolean** isFinal) {

**this**.isFinal = isFinal;

}

}