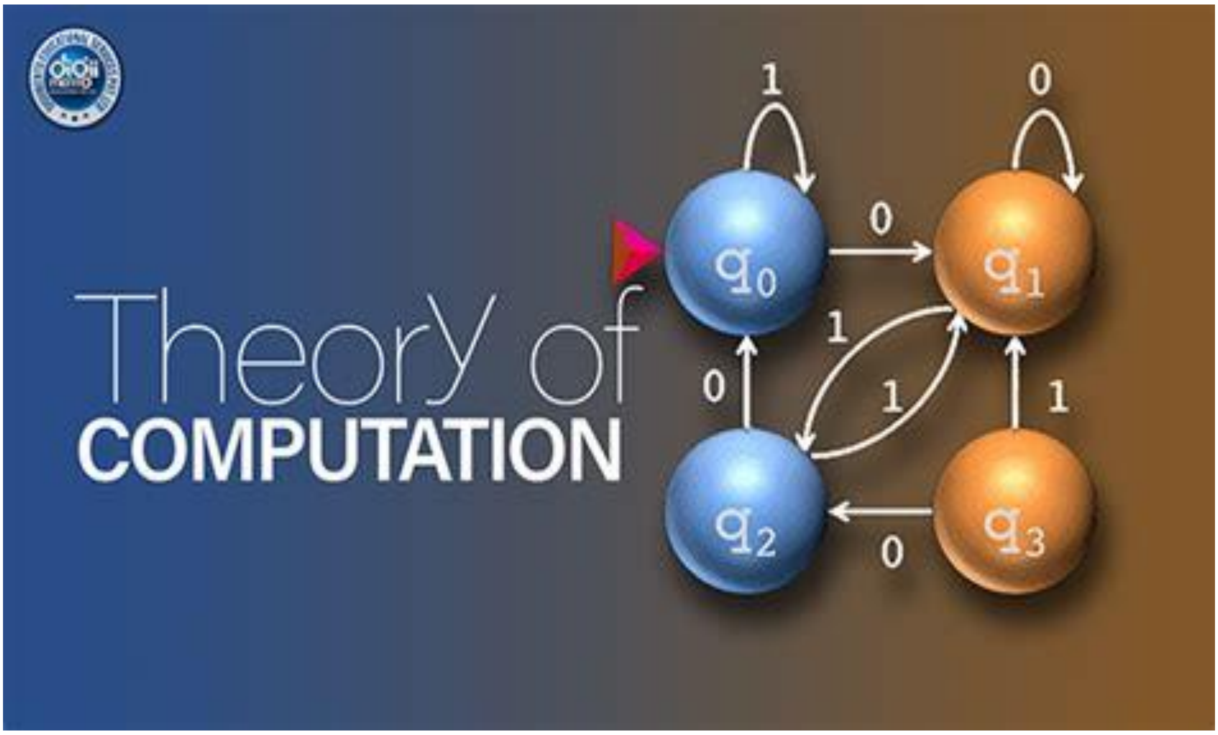
UNIVERSITY OF KWAZULU-NATAL ™

INYUVESI YAKWAZULU-NATALI

COMP314 – Theory of Computation

Assignment 1 – Group Y

218 067 924	**Hakeem Hoopdeo**
217 047 781	Kyle Naidoo
217 047 727	Thembelihle Mpofana

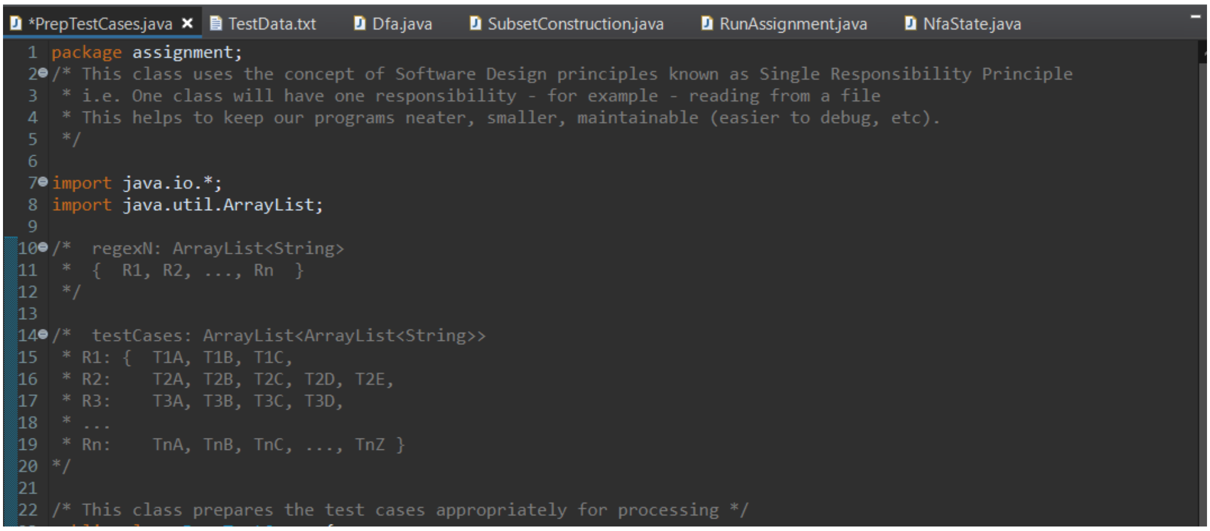


Introduction:

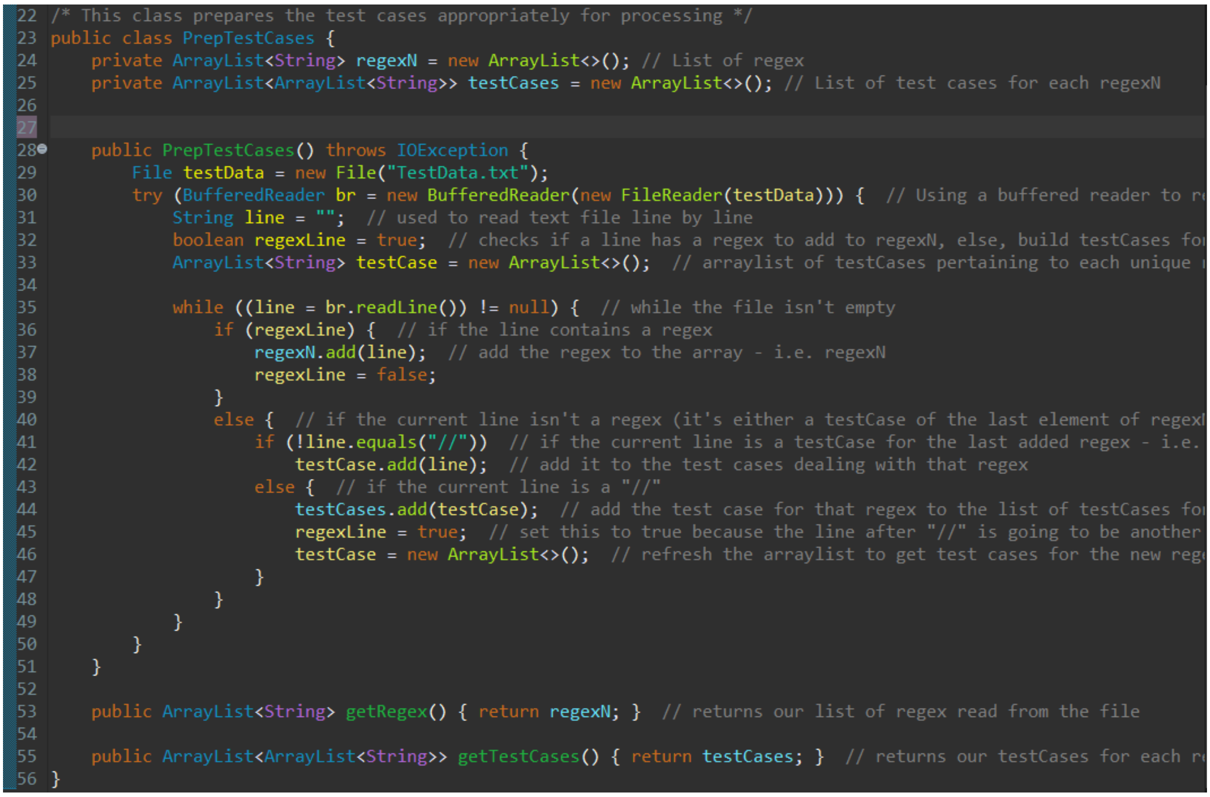
The purpose of this project was to work as a group and write a program that, given an NFA that was created using Thompson’s construction on a regular expression, write code to:

1. Read input from the *testData* file
2. Perform the Subset construction on the NFA to produce a DFA
3. Check if the testData pertaining to the particular DFA is accepted by the language defined by the DFA – D-RECOGNIZE algorithm
4. Run Assignment

1. Reading from the file:

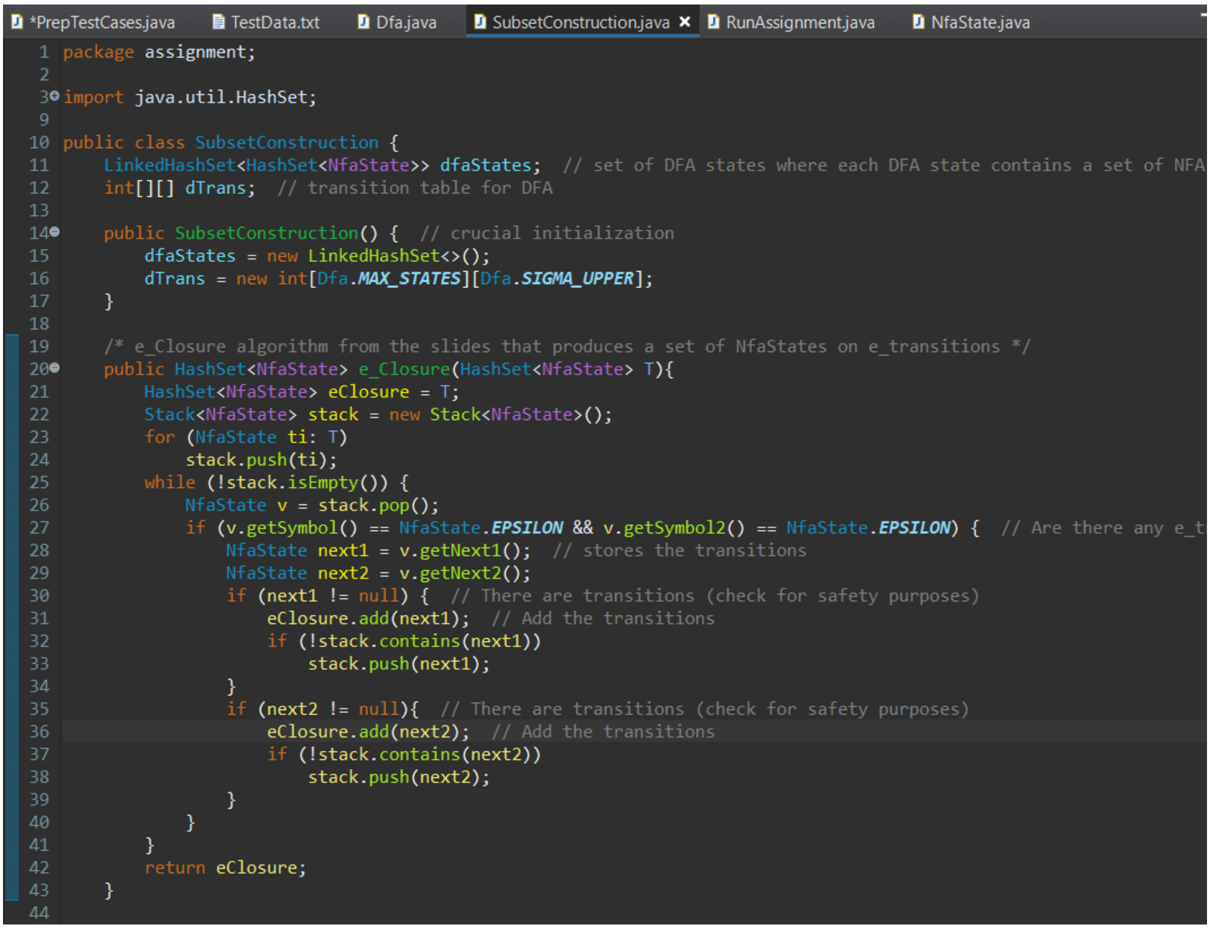


```java
package assignment;
/* This class uses the concept of Software Design principles known as Single Responsibility Principle
 * i.e. One class will have one responsibility - for example - reading from a file
 * This helps to keep our programs neater, smaller, maintainable (easier to debug, etc).
 */

import java.io.*;
import java.util.ArrayList;

/*  regexN: ArrayList<String>
 *  {  R1, R2, ..., Rn  }
 */

/*  testCases: ArrayList<ArrayList<String>>
 * R1: {  T1A, T1B, T1C,
 * R2:    T2A, T2B, T2C, T2D, T2E,
 * R3:    T3A, T3B, T3C, T3D,
 * ...
 * Rn:    TnA, TnB, TnC, ..., TnZ }
*/

/* This class prepares the test cases appropriately for processing */
```

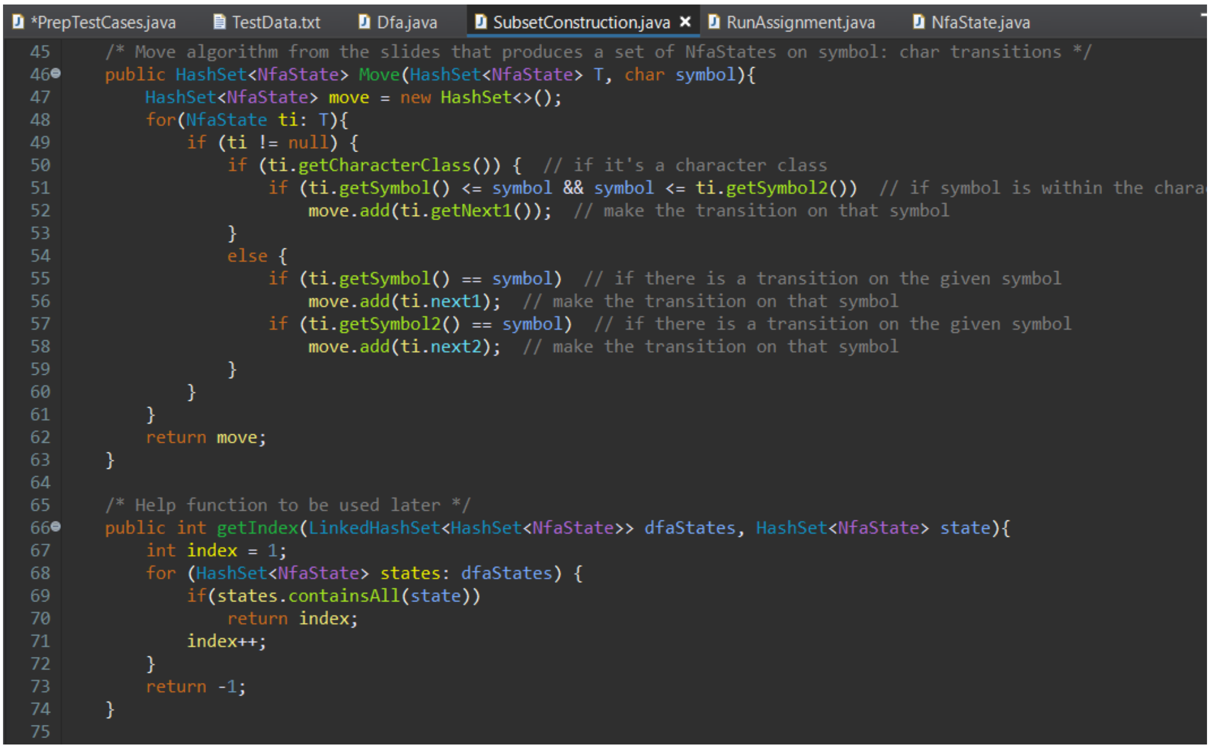


```java
/* This class prepares the test cases appropriately for processing */
public class PrepTestCases {
    private ArrayList<String> regexN = new ArrayList<>(); // List of regex
    private ArrayList<ArrayList<String>> testCases = new ArrayList<>(); // List of test cases for each regexN


    public PrepTestCases() throws IOException {
        File testData = new File("TestData.txt");
        try (BufferedReader br = new BufferedReader(new FileReader(testData))) {  // Using a buffered reader to re
            String line = "";  // used to read text file line by line
            boolean regexLine = true;  // checks if a line has a regex to add to regexN, else, build testCases fo
            ArrayList<String> testCase = new ArrayList<>();  // arraylist of testCases pertaining to each unique

            while ((line = br.readLine()) != null) {  // while the file isn't empty
                if (regexLine) {  // if the line contains a regex
                    regexN.add(line);  // add the regex to the array - i.e. regexN
                    regexLine = false;
                }
                else {  // if the current line isn't a regex (it's either a testCase of the last element of regexN
                    if (!line.equals("//"))  // if the current line is a testCase for the last added regex - i.e.
                        testCase.add(line);  // add it to the test cases dealing with that regex
                    else {  // if the current line is a "//"
                        testCases.add(testCase);  // add the test case for that regex to the list of testCases fo
                        regexLine = true;  // set this to true because the line after "//" is going to be another
                        testCase = new ArrayList<>();  // refresh the arraylist to get test cases for the new reg
                    }
                }
            }
        }
    }

    public ArrayList<String> getRegex() { return regexN; }  // returns our list of regex read from the file

    public ArrayList<ArrayList<String>> getTestCases() { return testCases; }  // returns our testCases for each r
}
```

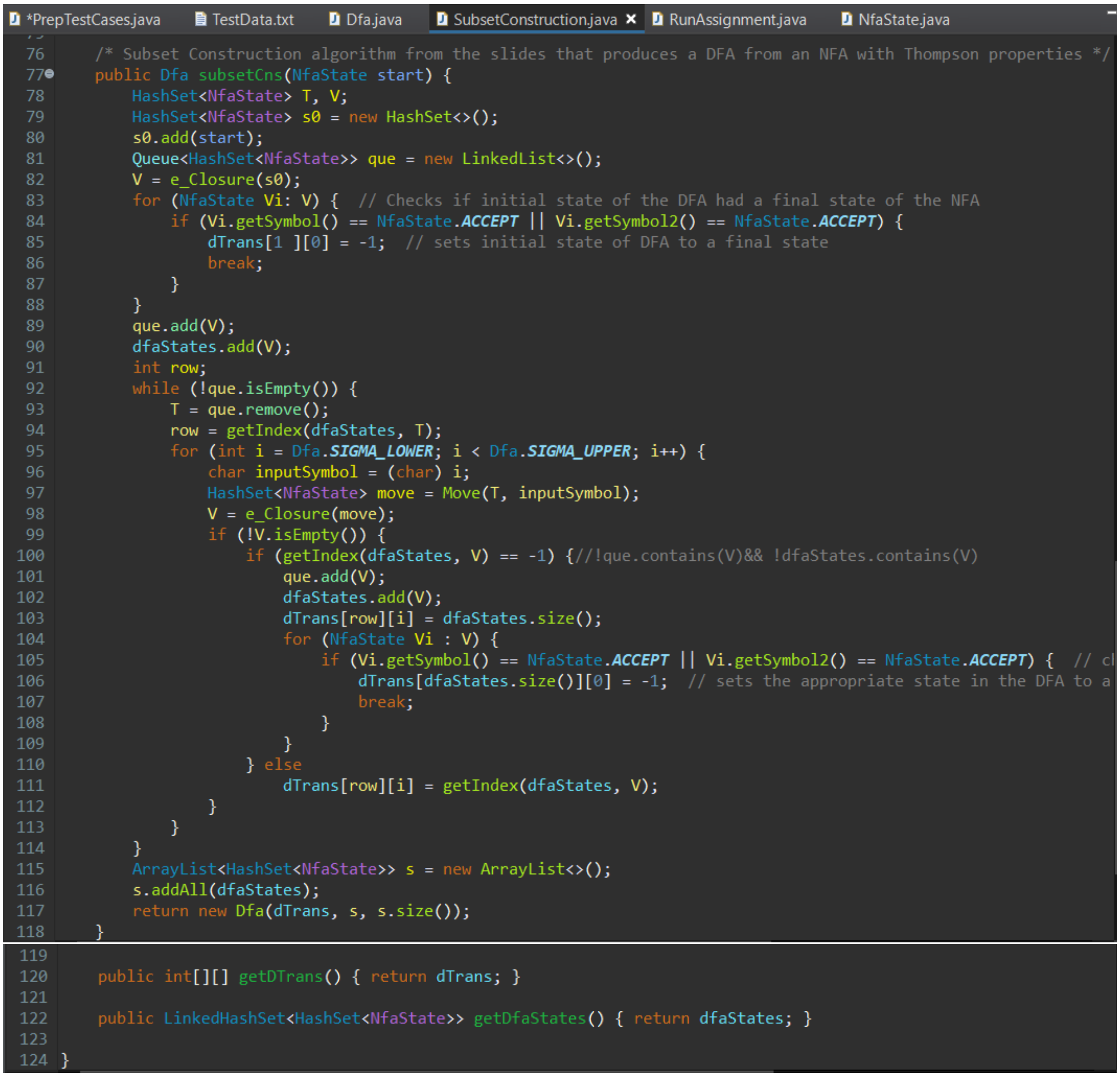
2. Perform Subset Construction:



```java
package assignment;

import java.util.HashSet;

public class SubsetConstruction {
    LinkedHashSet<HashSet<NfaState>> dfaStates;  // set of DFA states where each DFA state contains a set of NFA
    int[][] dTrans;  // transition table for DFA

    public SubsetConstruction() {  // crucial initialization
        dfaStates = new LinkedHashSet<>();
        dTrans = new int[Dfa.MAX_STATES][Dfa.SIGMA_UPPER];
    }

    /* e_Closure algorithm from the slides that produces a set of NfaStates on e_transitions */
    public HashSet<NfaState> e_Closure(HashSet<NfaState> T){
        HashSet<NfaState> eClosure = T;
        Stack<NfaState> stack = new Stack<NfaState>();
        for (NfaState ti: T)
            stack.push(ti);
        while (!stack.isEmpty()) {
            NfaState v = stack.pop();
            if (v.getSymbol() == NfaState.EPSILON && v.getSymbol2() == NfaState.EPSILON) {  // Are there any e_t
                NfaState next1 = v.getNext1();  // stores the transitions
                NfaState next2 = v.getNext2();
                if (next1 != null) {  // There are transitions (check for safety purposes)
                    eClosure.add(next1);  // Add the transitions
                    if (!stack.contains(next1))
                        stack.push(next1);
                }
                if (next2 != null){  // There are transitions (check for safety purposes)
                    eClosure.add(next2);  // Add the transitions
                    if (!stack.contains(next2))
                        stack.push(next2);
                }
            }
        }
        return eClosure;
    }

```

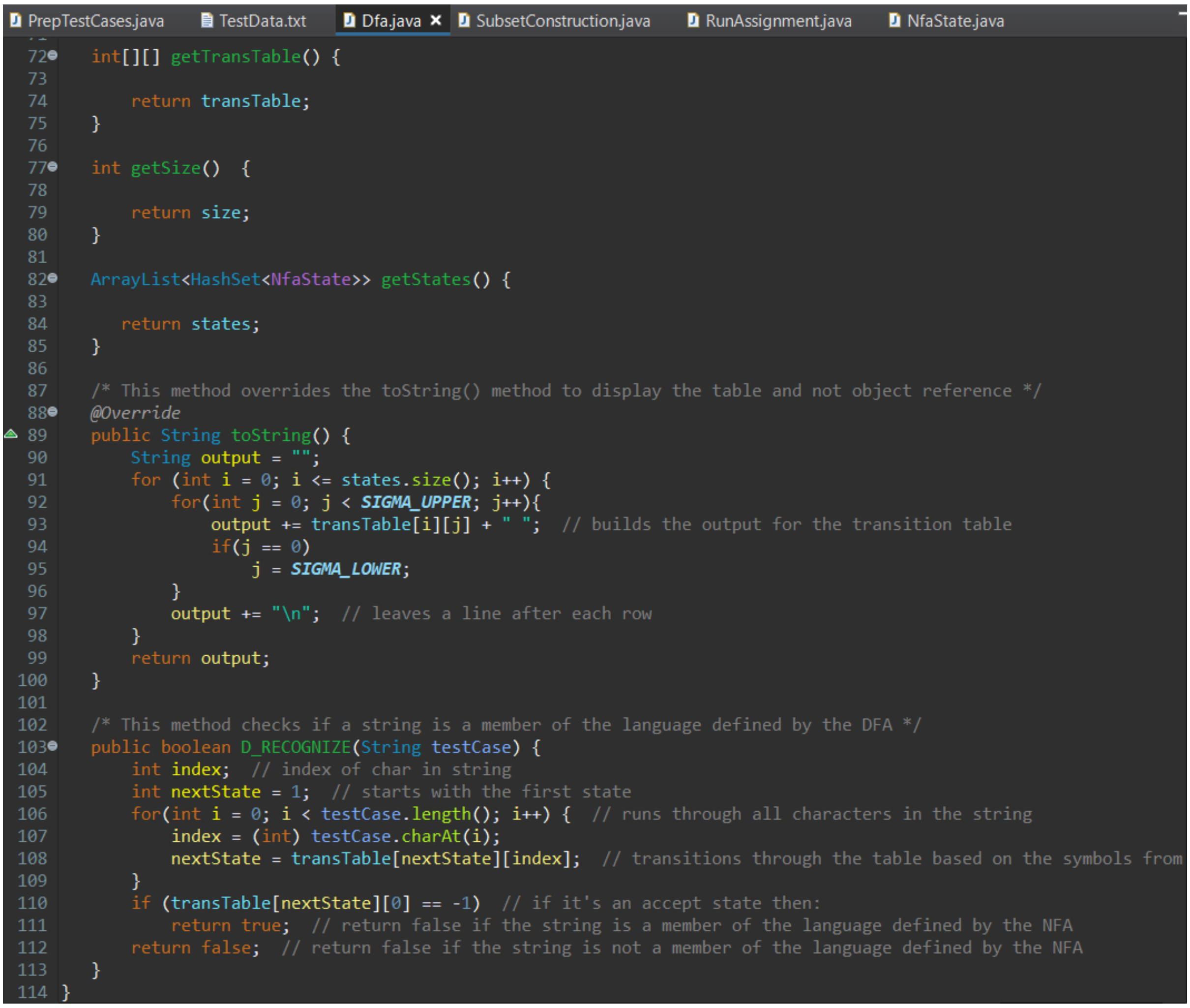


```java
    /* Move algorithm from the slides that produces a set of NfaStates on symbol: char transitions */
    public HashSet<NfaState> Move(HashSet<NfaState> T, char symbol){
        HashSet<NfaState> move = new HashSet<>();
        for(NfaState ti: T){
            if (ti != null) {
                if (ti.getCharacterClass()) {  // if it's a character class
                    if (ti.getSymbol() <= symbol && symbol <= ti.getSymbol2())  // if symbol is within the chara
                        move.add(ti.getNext1());  // make the transition on that symbol
                }
                else {
                    if (ti.getSymbol() == symbol)  // if there is a transition on the given symbol
                        move.add(ti.next1);  // make the transition on that symbol
                    if (ti.getSymbol2() == symbol)  // if there is a transition on the given symbol
                        move.add(ti.next2);  // make the transition on that symbol
                }
            }
        }
        return move;
    }

    /* Help function to be used later */
    public int getIndex(LinkedHashSet<HashSet<NfaState>> dfaStates, HashSet<NfaState> state){
        int index = 1;
        for (HashSet<NfaState> states: dfaStates) {
            if(states.containsAll(state))
                return index;
            index++;
        }
        return -1;
    }

```



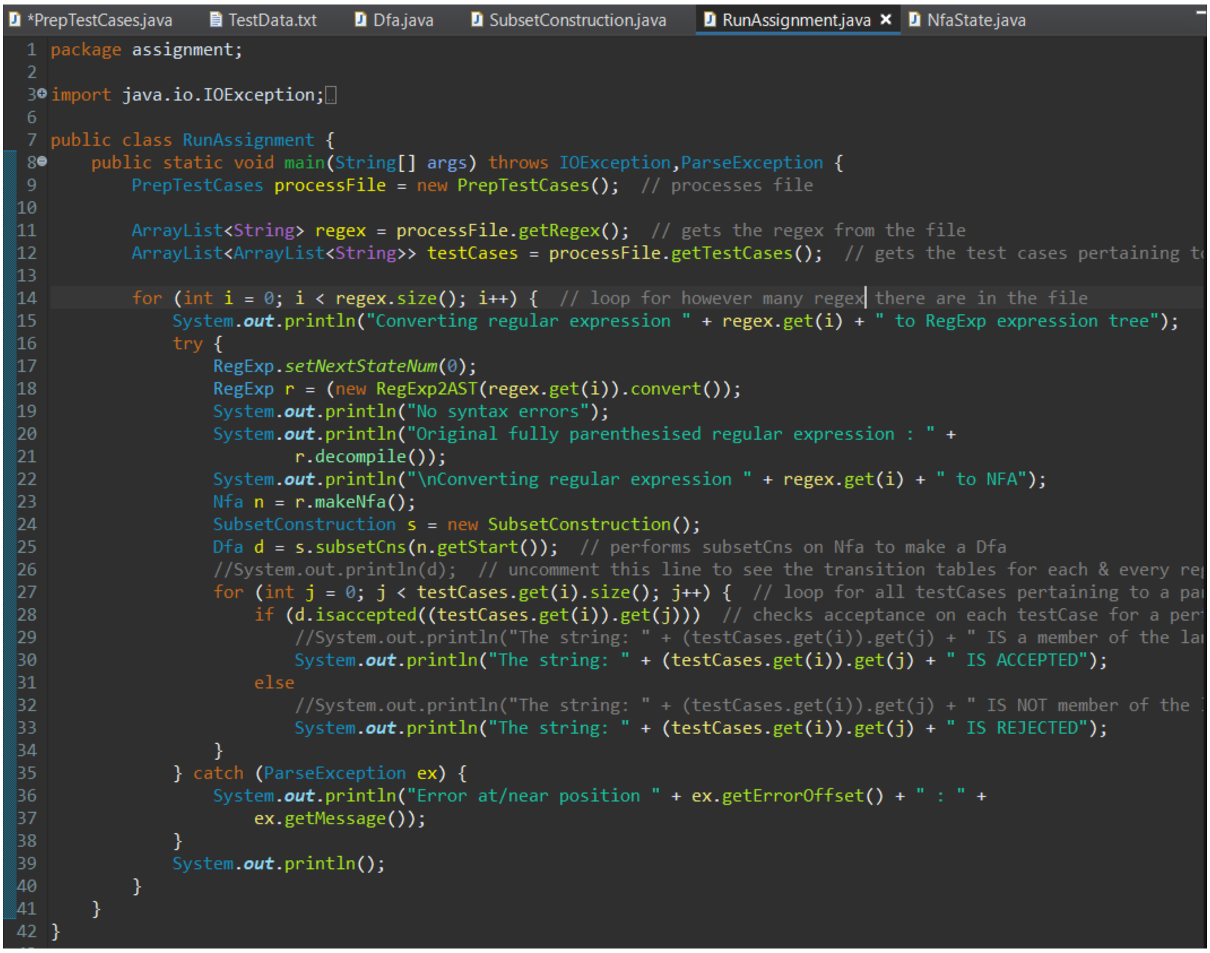
```java
    /* Subset Construction algorithm from the slides that produces a DFA from an NFA with Thompson properties */
    public Dfa subsetCns(NfaState start) {
        HashSet<NfaState> T, V;
        HashSet<NfaState> s0 = new HashSet<>();
        s0.add(start);
        Queue<HashSet<NfaState>> que = new LinkedList<>();
        V = e_Closure(s0);
        for (NfaState Vi: V) {  // Checks if initial state of the DFA had a final state of the NFA
            if (Vi.getSymbol() == NfaState.ACCEPT || Vi.getSymbol2() == NfaState.ACCEPT) {
                dTrans[1 ][0] = -1;  // sets initial state of DFA to a final state
                break;
            }
        }
        que.add(V);
        dfaStates.add(V);
        int row;
        while (!que.isEmpty()) {
            T = que.remove();
            row = getIndex(dfaStates, T);
            for (int i = Dfa.SIGMA_LOWER; i < Dfa.SIGMA_UPPER; i++) {
                char inputSymbol = (char) i;
                HashSet<NfaState> move = Move(T, inputSymbol);
                V = e_Closure(move);
                if (!V.isEmpty()) {
                    if (getIndex(dfaStates, V) == -1) {//!que.contains(V)&& !dfaStates.contains(V)
                        que.add(V);
                        dfaStates.add(V);
                        dTrans[row][i] = dfaStates.size();
                        for (NfaState Vi : V) {
                            if (Vi.getSymbol() == NfaState.ACCEPT || Vi.getSymbol2() == NfaState.ACCEPT) {  // cl
                                dTrans[dfaStates.size()][0] = -1;  // sets the appropriate state in the DFA to a
                                break;
                            }
                        }
                    } else
                        dTrans[row][i] = getIndex(dfaStates, V);
                }
            }
        }
        ArrayList<HashSet<NfaState>> s = new ArrayList<>();
        s.addAll(dfaStates);
        return new Dfa(dTrans, s, s.size());
    }

    public int[][] getDTrans() { return dTrans; }

    public LinkedHashSet<HashSet<NfaState>> getDfaStates() { return dfaStates; }

}
```

3. D-RECOGNIZE Algorithm for DFA:



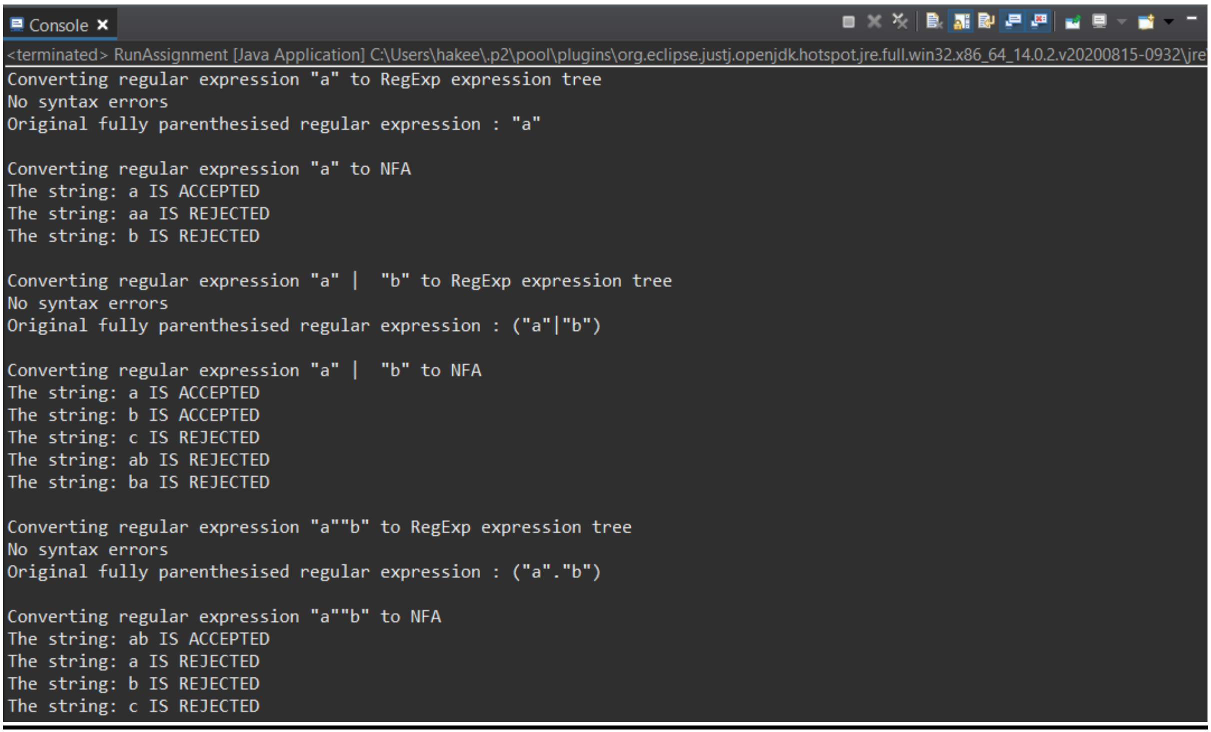
```java
    int[][] getTransTable() {

        return transTable;
    }

    int getSize()  {

        return size;
    }

    ArrayList<HashSet<NfaState>> getStates() {

       return states;
    }

    /* This method overrides the toString() method to display the table and not object reference */
    @Override
    public String toString() {
        String output = "";
        for (int i = 0; i <= states.size(); i++) {
            for(int j = 0; j < SIGMA_UPPER; j++){
                output += transTable[i][j] + " ";  // builds the output for the transition table
                if(j == 0)
                    j = SIGMA_LOWER;
            }
            output += "\n";  // leaves a line after each row
        }
        return output;
    }

    /* This method checks if a string is a member of the language defined by the DFA */
    public boolean D_RECOGNIZE(String testCase) {
        int index;  // index of char in string
        int nextState = 1;  // starts with the first state
        for(int i = 0; i < testCase.length(); i++) {  // runs through all characters in the string
            index = (int) testCase.charAt(i);
            nextState = transTable[nextState][index];  // transitions through the table based on the symbols from
        }
        if (transTable[nextState][0] == -1)  // if it's an accept state then:
            return true;  // return false if the string is a member of the language defined by the NFA
        return false;  // return false if the string is not a member of the language defined by the NFA
    }
}
```

4. RunAssignment:

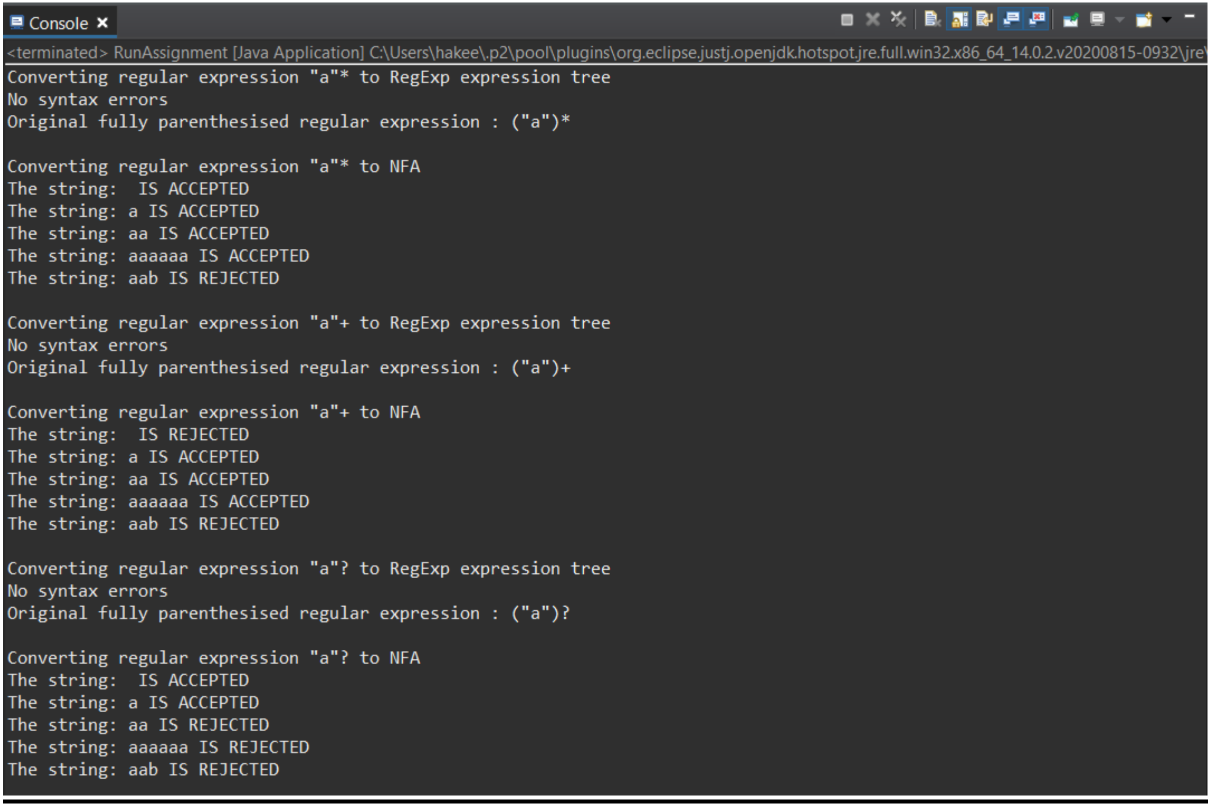


```java
package assignment;

import java.io.IOException;

public class RunAssignment {
    public static void main(String[] args) throws IOException,ParseException {
        PrepTestCases processFile = new PrepTestCases();  // processes file

        ArrayList<String> regex = processFile.getRegex();  // gets the regex from the file
        ArrayList<ArrayList<String>> testCases = processFile.getTestCases();  // gets the test cases pertaining t

        for (int i = 0; i < regex.size(); i++) {  // loop for however many regex there are in the file
            System.out.println("Converting regular expression " + regex.get(i) + " to RegExp expression tree");
            try {
                RegExp.setNextStateNum(0);
                RegExp r = (new RegExp2AST(regex.get(i)).convert());
                System.out.println("No syntax errors");
                System.out.println("Original fully parenthesised regular expression : " +
                        r.decompile());
                System.out.println("\nConverting regular expression " + regex.get(i) + " to NFA");
                Nfa n = r.makeNfa();
                SubsetConstruction s = new SubsetConstruction();
                Dfa d = s.subsetCns(n.getStart());  // performs subsetCns on Nfa to make a Dfa
                //System.out.println(d);  // uncomment this line to see the transition tables for each & every re
                for (int j = 0; j < testCases.get(i).size(); j++) {  // loop for all testCases pertaining to a pa
                    if (d.isaccepted((testCases.get(i)).get(j)))  // checks acceptance on each testCase for a per
                        //System.out.println("The string: " + (testCases.get(i)).get(j) + " IS a member of the la
                        System.out.println("The string: " + (testCases.get(i)).get(j) + " IS ACCEPTED");
                    else
                        //System.out.println("The string: " + (testCases.get(i)).get(j) + " IS NOT member of the
                        System.out.println("The string: " + (testCases.get(i)).get(j) + " IS REJECTED");
                }
            } catch (ParseException ex) {
                System.out.println("Error at/near position " + ex.getErrorOffset() + " : " +
                    ex.getMessage());
            }
            System.out.println();
        }
    }
}
```

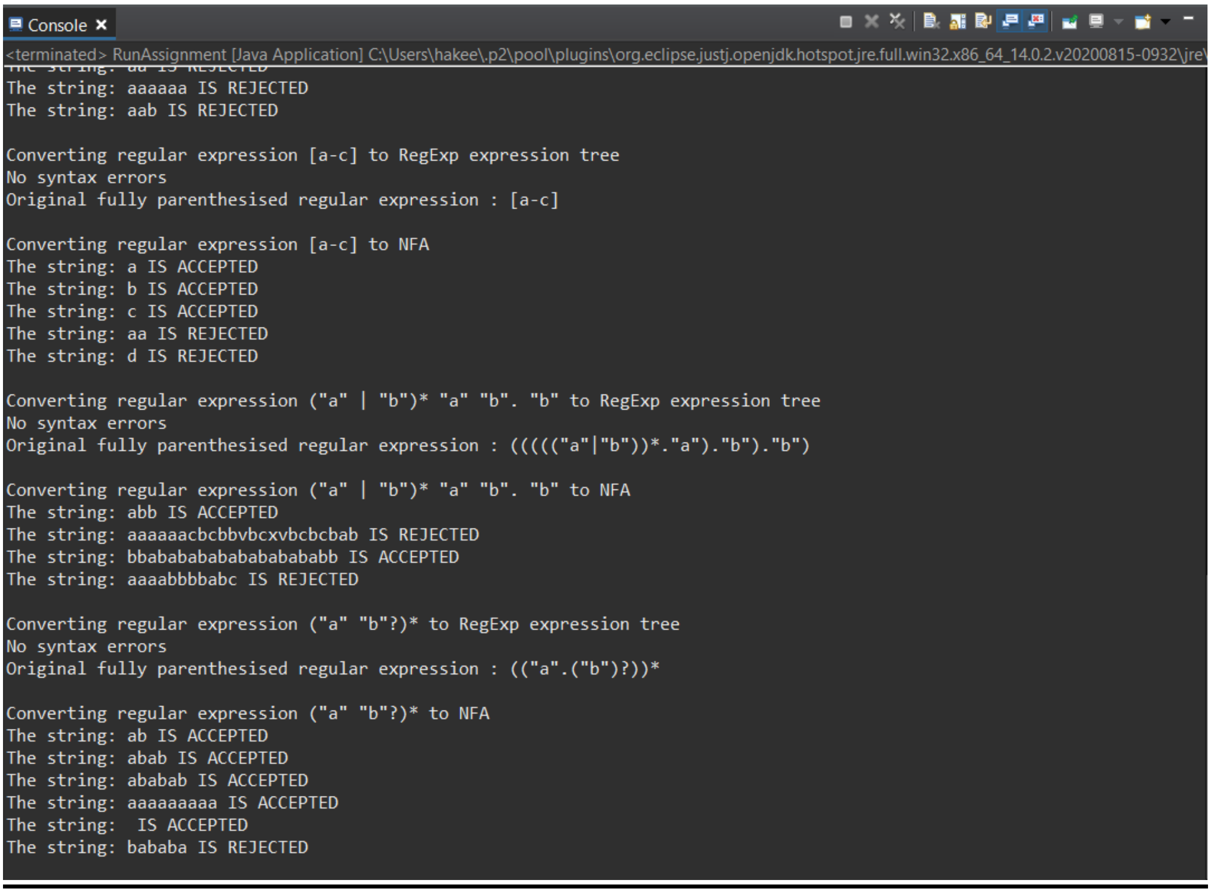
Testing Record depicting Code Correctness:



```
<terminated> RunAssignment [Java Application] C:\Users\hakee\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_14.0.2.v20200815-0932\jre
Converting regular expression "a" to RegExp expression tree
No syntax errors
Original fully parenthesised regular expression : "a"

Converting regular expression "a" to NFA
The string: a IS ACCEPTED
The string: aa IS REJECTED
The string: b IS REJECTED

Converting regular expression "a" |  "b" to RegExp expression tree
No syntax errors
Original fully parenthesised regular expression : ("a"|"b")

Converting regular expression "a" |  "b" to NFA
The string: a IS ACCEPTED
The string: b IS ACCEPTED
The string: c IS REJECTED
The string: ab IS REJECTED
The string: ba IS REJECTED

Converting regular expression "a""b" to RegExp expression tree
No syntax errors
Original fully parenthesised regular expression : ("a"."b")

Converting regular expression "a""b" to NFA
The string: ab IS ACCEPTED
The string: a IS REJECTED
The string: b IS REJECTED
The string: c IS REJECTED
```



```
<terminated> RunAssignment [Java Application] C:\Users\hakee\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_14.0.2.v20200815-0932\jre\
Converting regular expression "a"* to RegExp expression tree
No syntax errors
Original fully parenthesised regular expression : ("a")*

Converting regular expression "a"* to NFA
The string:  IS ACCEPTED
The string: a IS ACCEPTED
The string: aa IS ACCEPTED
The string: aaaaaa IS ACCEPTED
The string: aab IS REJECTED

Converting regular expression "a"+ to RegExp expression tree
No syntax errors
Original fully parenthesised regular expression : ("a")+

Converting regular expression "a"+ to NFA
The string:  IS REJECTED
The string: a IS ACCEPTED
The string: aa IS ACCEPTED
The string: aaaaaa IS ACCEPTED
The string: aab IS REJECTED

Converting regular expression "a"? to RegExp expression tree
No syntax errors
Original fully parenthesised regular expression : ("a")?

Converting regular expression "a"? to NFA
The string:  IS ACCEPTED
The string: a IS ACCEPTED
The string: aa IS REJECTED
The string: aaaaaa IS REJECTED
The string: aab IS REJECTED
```



```
<terminated> RunAssignment [Java Application] C:\Users\hakee\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_14.0.2.v20200815-0932\jre\
The string: aa IS REJECTED
The string: aaaaaa IS REJECTED
The string: aab IS REJECTED

Converting regular expression [a-c] to RegExp expression tree
No syntax errors
Original fully parenthesised regular expression : [a-c]

Converting regular expression [a-c] to NFA
The string: a IS ACCEPTED
The string: b IS ACCEPTED
The string: c IS ACCEPTED
The string: aa IS REJECTED
The string: d IS REJECTED

Converting regular expression ("a" | "b")* "a" "b". "b" to RegExp expression tree
No syntax errors
Original fully parenthesised regular expression : ((((("a"|"b"))*."a")."b")."b")

Converting regular expression ("a" | "b")* "a" "b". "b" to NFA
The string: abb IS ACCEPTED
The string: aaaaaacbcbbvbcxvbcbcbab IS REJECTED
The string: bbabababababababababb IS ACCEPTED
The string: aaaabbbbabc IS REJECTED

Converting regular expression ("a" "b"?)* to RegExp expression tree
No syntax errors
Original fully parenthesised regular expression : (("a".("b")?))*

Converting regular expression ("a" "b"?)* to NFA
The string: ab IS ACCEPTED
The string: abab IS ACCEPTED
The string: ababab IS ACCEPTED
The string: aaaaaaaaa IS ACCEPTED
The string:  IS ACCEPTED
The string: bababa IS REJECTED
```

Contributions:

Before anything, I must say that I am very pleased with the work that my group has produced. Not only am I proud of my group members performance, but I am also grateful that I had the opportunity to work with these guys. Without their hard work, dedication, and overall superb team effort, we would not have accomplished *perfection.*

NB: As Group Leader of this project, I had my hands in all parts of this project.

Classes Written:

1. Hakeem: PrepTestCases.java, RunAssignment.java
2. Kyle: Dfa.java: toString(), D_RECOGNIZE
3. Thembelihle: SubsetConstruction.java

Bug fixes:

1. Hakeem: SubsetConstruction.java: e_Closure, Move, subsetCns, Dfa.java: D_RECOGNIZE

Testing:

1. Kyle: Extensively tested D_RECOGNIZE
2. Thembelihle: Extensively tested e_Closure, Move, subsetCns
3. Hakeem: Tested the overall program – RunAssignment.java

Final Takings on Contributions:

I believe that Kyle, Thembelihle and myself deserve the same marks since we worked together, partitioned the workload, scheduled zoom meetings for discussions and fixes and overall have shown excellent team synergy. This assignment has also gone through extensive testing and works for more than just the test cases provided in the file. Lastly:

** Siyavuya Ngalonkulu – 216 035 051: contributed nothing towards assignment 1. He therefore has my highest recommendations to receive a "*0*" mark for this assignment. This recommendation is unbiased, and the rest of the group share a similar sentiment towards this issue.

Conclusion:

My group and I found this assignment to be very enjoyable. We had fun trying to figure out how to translate the pseudocode into actual code and thoroughly enjoyed working with the data structures rich with object-oriented programming design. It gave use a reminder of the good old days of second year and helped us brush up on our skills. It also allowed us to see that these concepts we learned in second year have limitless applications. We therefore collectively thank you for this assignment and the knowledge, experience and insight gained from it!