PROGRAM:

```
import java.util.*;
public class nfaTodfa {
   public static void print(List<List<List<Integer>>> table) {
      System.out.print("  STATE/INPUT  |");
      char a = 'a';
      for (int i = 0; i < table.get(0).size() - 1; i++) {
         System.out.print("   " + a++ + "   |");
      }
      System.out.println("   ^   ");
      System.out.println();
      for (int i = 0; i < table.size(); i++) {
         System.out.print("       " + i + "      ");
         for (int j = 0; j < table.get(i).size(); j++) {
            System.out.print(" | ");
            for (int k = 0; k < table.get(i).get(j).size(); k++) {
               System.out.print(table.get(i).get(j).get(k) + " ");
            }
         }
         System.out.println();
      }
   }

   public static void printdfa(List<List<Integer>> states, List<List<List<Integer>>>
dfa) {
      System.out.print("  STATE/INPUT  ");
      char a = 'a';
```

```
    for (int i = 0; i < dfa.get(0).size(); i++) {
        System.out.print("|   " + a++ + "   ");
    }
    System.out.println();
    for (int i = 0; i < states.size(); i++) {
        System.out.print("{ ");
        for (int h = 0; h < states.get(i).size(); h++)
            System.out.print(states.get(i).get(h) + " ");
        if (states.get(i).isEmpty()) {
            System.out.print("^ ");
        }
        System.out.print("} ");
        for (int j = 0; j < dfa.get(i).size(); j++) {
            System.out.print(" | ");
            for (int k = 0; k < dfa.get(i).get(j).size(); k++) {
                System.out.print(dfa.get(i).get(j).get(k) + " ");
            }
            if (dfa.get(i).get(j).isEmpty()) {
                System.out.print("^ ");
            }
        }
        System.out.println();
    }
}

public static List<Integer> closure(int s, List<List<List<Integer>>> v) {
```

```
    List<Integer> t = new ArrayList<>();
    Queue<Integer> q = new LinkedList<>();
    t.add(s);
    int a = v.get(s).get(v.get(s).size() - 1).size();
    for (int i = 0; i < a; i++) {
        t.add(v.get(s).get(v.get(s).size() - 1).get(i));
        q.add(t.get(i));
    }
    while (!q.isEmpty()) {
        int f = q.poll();
        if (!v.get(f).get(v.get(f).size() - 1).isEmpty()) {
            int u = v.get(f).get(v.get(f).size() - 1).size();
            for (int i = 0; i < u; i++) {
                int y = v.get(f).get(v.get(f).size() - 1).get(i);
                if (!t.contains(y)) {
                    t.add(y);
                    q.add(y);
                }
            }
        }
    }
    return t;
}

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
```

```
        int n, alpha;
        System.out.println("************************* NFA to DFA *************************");
        System.out.println();
        System.out.print("Enter total number of states in NFA: ");
        n = sc.nextInt();
        System.out.print("Enter number of elements in alphabet: ");
        alpha = sc.nextInt();
        List<List<List<Integer>>> table = new ArrayList<>();
        for (int i = 0; i < n; i++) {
            System.out.println("For state " + i);
            List<List<Integer>> v = new ArrayList<>();
            char a = 'a';
            int y, yn;
            for (int j = 0; j < alpha; j++) {
                List<Integer> t = new ArrayList<>();
                System.out.print("Enter no. of output states for input " + a++ + ": ");
                yn = sc.nextInt();
                System.out.println("Enter output states:");
                for (int k = 0; k < yn; k++) {
                    y = sc.nextInt();
                    t.add(y);
                }
                v.add(t);
            }
            List<Integer> t = new ArrayList<>();
            System.out.print("Enter no. of output states for input ^: ");
```

```
    yn = sc.nextInt();
    System.out.println("Enter output states:");
    for (int k = 0; k < yn; k++) {
        y = sc.nextInt();
        t.add(y);
    }
    v.add(t);
    table.add(v);
}
System.out.println("***** TRANSITION TABLE OF NFA *****");
print(table);
System.out.println();
System.out.println("***** TRANSITION TABLE OF DFA *****");
List<List<List<Integer>>> dfa = new ArrayList<>();
List<List<Integer>> states = new ArrayList<>();
states.add(closure(0, table));
Queue<List<Integer>> q = new LinkedList<>();
q.add(states.get(0));
while (!q.isEmpty()) {
    List<Integer> f = q.poll();
    List<List<Integer>> v = new ArrayList<>();
    for (int i = 0; i < alpha; i++) {
        List<Integer> t = new ArrayList<>();
        Set<Integer> s = new HashSet<>();
        for (int j = 0; j < f.size(); j++) {
            for (int k = 0; k < table.get(f.get(j)).get(i).size(); k++) {
```

```
                List<Integer> cl = closure(table.get(f.get(j)).get(i).get(k), table);
                s.addAll(cl);
              }
            }
            t.addAll(s);
            v.add(t);
            if (!states.contains(t)) {
              states.add(t);
              q.add(t);
            }
          }
          dfa.add(v);
        }
        printdfa(states, dfa);
        sc.close();
    }
}
```

OUTPUT:

```
Enter total number of states in NFA : 4
Enter number of elements in alphabet : 2
For state 0
Enter no. of output states for input a : 1
Enter output states :
0
Enter no. of output states for input b : 2
Enter output states :
0
1
Enter no. of output states for input ^ : 0
Enter output states :
For state 1
Enter no. of output states for input a : 1
Enter output states :
2
Enter no. of output states for input b : 0
Enter output states :
Enter no. of output states for input ^ : 0
Enter output states :
For state 2
Enter no. of output states for input a : 0
Enter output states :
Enter no. of output states for input b : 1
Enter output states :
3
Enter no. of output states for input ^ : 0
Enter output states :
For state 3
Enter no. of output states for input a : 0
Enter output states :
Enter no. of output states for input b : 0
Enter output states :
Enter no. of output states for input ^ : 0
Enter output states :
***** TRANSITION TABLE OF NFA *****
  STATE/INPUT  |   a   |   b   |   ^

       0       | 0  | 0 1  |
       1       | 2  |  |
       2       |  | 3  |
       3       |  |  |

***** TRANSITION TABLE OF DFA *****
  STATE/INPUT  |   a   |   b
{ 0 }  | 0  | 0 1
{ 0 1 }  | 0 2  | 0 1
{ 0 2 }  | 0  | 0 1 3
{ 0 1 3 }  | 0 2  | 0 1
```

OBSERVATION:

In this experiment, I learned about non deterministic finite automata(NFA), how to construct a deterministic finite automata (DFA) from the given non deterministic finite automata using subset construction method, Also I have learnt how to construct transition table for the DFA from the given NFA.

CONCLUSION:

Through this experiment, I gained a deeper understanding of non deterministic finite automata (NFA) in string validation and pattern recognition. By learning how to construct a DFA from given NFA, I acquired the necessary skills to design and implement NFAs for various scenarios.