Experiment VII

**Aim**: Write a program to minimize any given DFA.

Algorithm

1. Start
2. Load and display the DFA transition table.
3. Divide DFA into final and non-final states.
4. For the equivalent class, find the new DFA.
5. Equivalent classes are segmented to get new equivalent classes.
6. Repeat the process until the DFA cannot be minimized further.
7. Find the new final states of the minimized DFA.
8. Display the transition table of the minimized DFA.
9. Stop

Output

DFA: STATE TRANSITION TABLE

| 0 1

-----+------------

A | B C

B | E F

C | A A

D | F E

E | D F

F | D E

Final states = EF

EQUIV. CLASS CANDIDATE ==> 0:[ABCD] 1:[EF]

0:[ABCD] --> [BEAF] (0101)

0:[ABCD] --> [CFAE] (0101)

1:[EF] --> [DD] (00)

1:[EF] --> [FE] (11)

EQUIV. CLASS CANDIDATE ==> 0:[AC] 1:[BD] 2:[EF]

0:[AC] --> [BA] (10)

0:[AC] --> [CA] (00)

1:[BD] --> [EF] (22)

1:[BD] --> [FE] (22)

2:[EF] --> [DD] (11)

2:[EF] --> [FE] (22)

EQUIV. CLASS CANDIDATE ==> 0:[A] 1:[BD] 2:[C] 3:[EF]

0:[A] --> [B] (1)

0:[A] --> [C] (2)

1:[BD] --> [EF] (33)

1:[BD] --> [FE] (33)

2:[C] --> [A] (0)

2:[C] --> [A] (0)

3:[EF] --> [DD] (11)

3:[EF] --> [FE] (33)

DFA: STATE TRANSITION TABLE

| 0 1

-----+------------

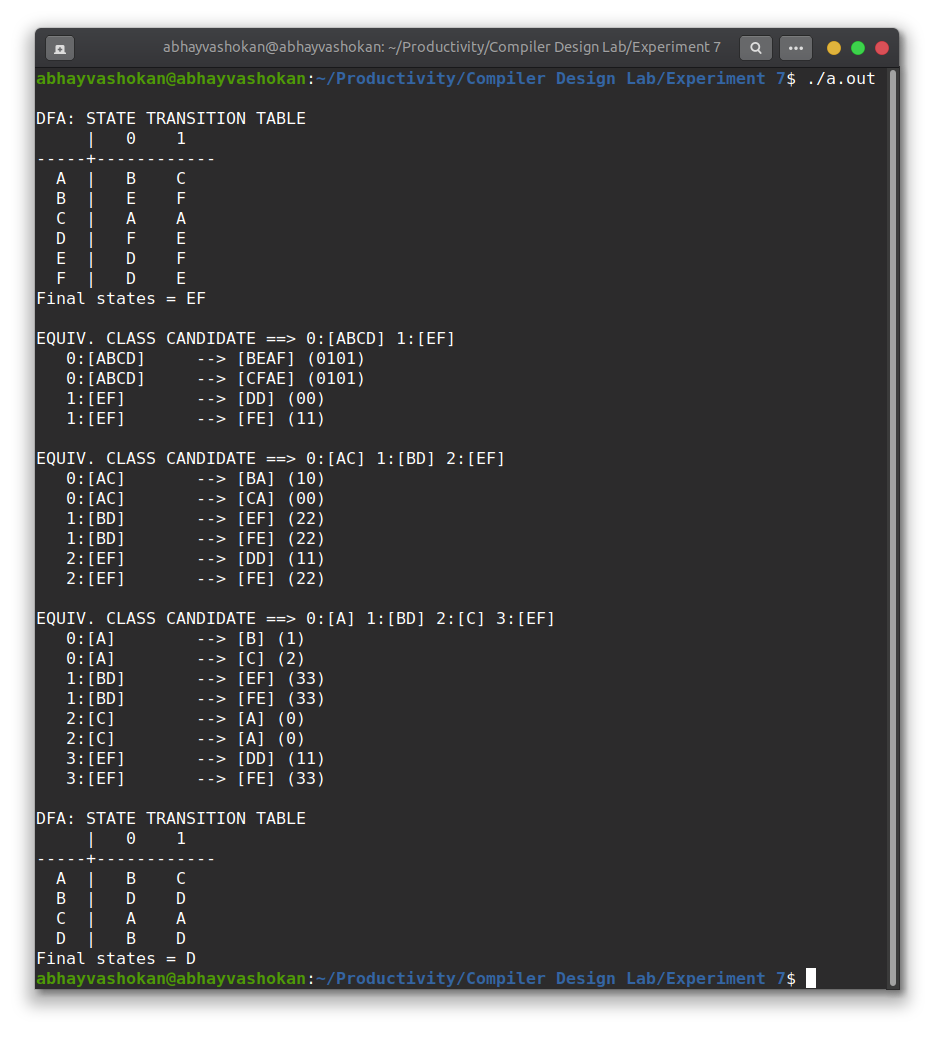
A | B C

B | D D

C | A A

D | B D

Final states = D

Screenshot

Readme

1. Compile and run the C program using the command

**gcc 2Abhay-P7.c && ./a.out**

2. The minimized DFA of the given DFA will be obtained as output.

**Result**: Successfully implemented a program to minimize a DFA.