Programming Assignment #3
DUE April 23, 2017

You may complete this assignment either INDIVIDUALLY or in a TEAM of TWO.
(See special instructions at end of assignment for teams.)

Complete this programming assignment using a high-level procedural programming language (e.g., Java, C, C++; not MATLAB) of your choice. Carefully read the instructions below regarding printouts and other documentation; note that you do NOT submit an executable copy of your program.

The deliverables for this programming assignment must be submitted electronically under the "Assignments" tab on T-Square no later than 11:00 pm on the due date listed above. A single pdf file is the preferred format. Clearly PRINT your name(s) at the top of each sheet. You are responsible for ensuring that your submitted file(s) are readable.

This assignment requires you to implement a graph data structure for representing a finite state machine (FSM). This data structure will be populated using a series of inputs that define the nodes and transition arcs of the FSM and, then, the program will generate both a representation of the graph structure and a state table representation of the FSM.

You are expected to develop an appropriate data structure (or set of data structures) and the software routines to implement the required functions. The precise format and implementation details are up to you, subject to the constraints described below. While this assignment may be more naturally matched to a dynamically-allocated data structure, it is allowable to statically allocate memory space for the data structure. For the output, you may use any appropriate Mealy/Moore state table format.

SPECIFICATIONS

General:

- Both Mealy and Moore machines must be supported by the program
- Maximum number of nodes (states): 25. Each state is identified by a unique name consisting of 1-8 alphanumeric characters.
- Maximum number of input bits: 4
- Output values are specified as a string of 1-5 alphanumeric characters

Inputs:

- The program may either ask the user to enter the FSM configuration (number of states, number of input bits, and Mealy or Moore) or determine these values from the input statements.
- Data describing the FSM is entered using a series of statements in the following formats:

{Mealy}	{Moore}
NODE *name*	NODE *name* / *output*
ARC *fromNode toNode inputs* / *output*	ARC *fromNode toNode inputs*

Inputs are specified as a string of bits $X_1X_2\ldots X_k$, where each bit is 0, 1, or x (don't care).

These statements may be entered in any order subject to the constraint that nodes must be defined using a NODE statement before being referenced in an ARC statement. If an ARC statement refers to an undefined node, an error message should be generated and the statement ignored. No other input error checking is required; you may assume users will enter fields correctly.

Outputs:

- The program must generate the following two outputs. No specific formatting is specified.

  GRAPH

  This output lists all of the graph nodes, in alphabetical order by state name, with each state name listed on a separate line followed by the state output value (if Moore). Transition arcs leaving that node are then listed (in any order) on the following line(s) with the destination state name, the input value causing the transition, and the associated output value (if Mealy).

  For each node (state), determine if the outgoing transitions are either incompletely specified or conflicting. For example, if there are two input bits, then outgoing transitions should be specified for inputs 00, 01, 10, and 11. If any of these are not specified, then generate a warning message for incomplete specification. If a particular input value is specified more than once, generate an error message for conflicting transitions.

  TABLE

  This output consists of a state table for the Mealy or Moore FSM, in any appropriate format, with current states listed in alphabetical order by state name. If some of the state transitions are incompletely specified (as defined above), list "don't care" (x) values for the next state and, if a Mealy machine, the output. If a particular output transition has conflicting specifications, list "error".

Example:

- Following is a simple example for a Mealy FSM with two states and two input bits. The formats shown are not intended as required or recommended.

  NODE Blue
  NODE Red
  ARC Red Red 00 / hot { note that the output is not required to be numeric }
  ARC Blue Blue 0x / hold { note that input 0x defines both 00 and 01 }
  ARC Red Blue 01 / cold
  ARC Orange Blue 10 / cold
  % error: state "Orange" not defined %
  ARC Blue Red 11 / on
  ARC Red Blue 1x / off

- Output GRAPH:

  Red
  Red 00 / hot
  Blue 01 / cold
  Blue 1x / off
  Blue
  Red 11 / on
  Blue 0x / hold
  % warning: input 10 not specified %

- Output TABLE:

Current State	Next State / Ouput X = 00	X = 01	X = 10	X = 11
Red	Red / hot	Blue / cold	Blue / off	Blue /off
Blue	Blue / hold	Blue / hold	x / x	Red / on

Submit the following materials:

(A) A brief description of the data structure you defined and implemented. DO NOT simply copy lines from your program code, but clearly *describe the concept* of how you organized the FSM data using your graph data structure.

(B) A test plan consisting of a series of input sets that demonstrate the correct functioning of your program, including the ability to represent both Mealy and Moore machines and the correct implementation of error checking (as defined above) for undefined states in the inputs and incompletely specified or conflicting output transitions. The full test plan will require multiple test cases, some of which may require a separate execution of the program.

(C) A listing of your program source code (.pdf or .txt).

(D) The program output (captured from your computer screen or printed to a file for the test plan defined above in (B), demonstrating your program's correct execution (including both GRAPH and TABLE outputs) and annotated to show implementation of the required error checking.

SPECIAL INSTRUCTIONS for teams of two:

You are expected to work together to define the data structure, overall program operation, and the parameters/results for key functions and procedures. **One person should write the code for processing of the input statements and the other person should write the code for generating the two output formats.** In the initial program comments, identify who was responsible for each of these assignments.

You are expected to work together to develop the program test plan, defined above in (B), to test and debug the program, and to generate the report. Upload only one copy of the report to T-Square under either of your names, making sure that both names are listed on the report.