PROBLEM STATEMENT:

Given a five button character combination lock; write a program which will simulate the
behavior of the lock when five characters are entered. The actions are unlock (if correct
sequence entered) and an alarm (if the incorrect sequence is entered). A table ADT must be
used to store the transition table and action table of the finite state machine (FSM) which models the
behavior of the lock. The correct combination will be given in a file

PROBLEM BACKGROUND :

Some of the best known of all table-driven algorithms are based on a model known
as a finite state machine (FSM). A finite state machine has five parts:

> - A set of states
> - A start state (which is an element of the set of states)
> - A set of potential conditions (and/or events)
> - A transition function that defines how to progress from one state to
>   the next
> - An action function that associates an action with each transition
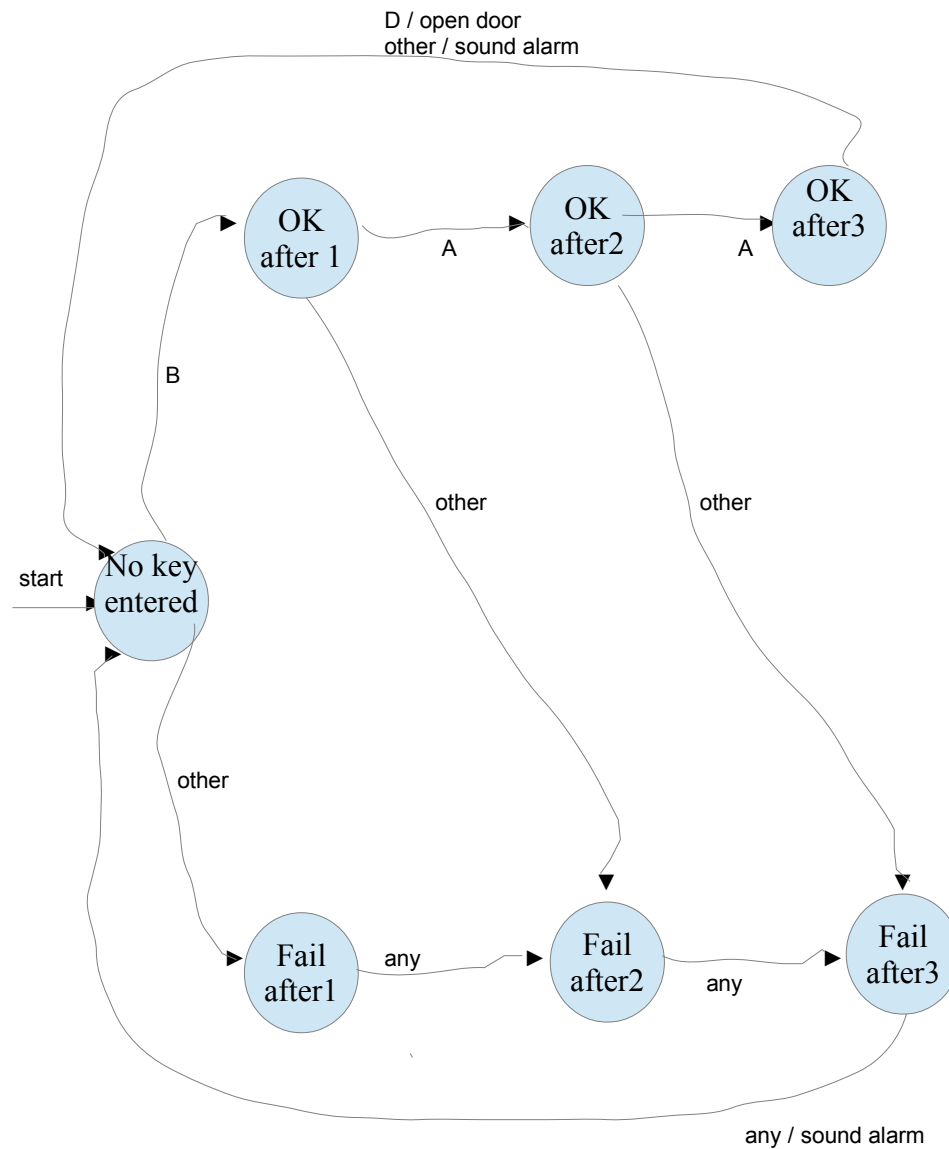
State machines are typically pictured in state diagram form as a group of
bubbles (one for each state) and arcs (one for each transition and associated action)
The transition arrow is labeled with a condition or event (or both) under which
such a transition occurs. This condition or event is the first part of any transition
label (to the left of a forward slash, "/", symbol).
Many state machines can be directly translated into highly efficient table-
driven algorithms. Such algorithms are based on the insight that transitions and
actions can be characterized as two tables. One of these tables, often called the
transition table, stores the transition function of the state machine. The other
table, sometimes called the action table, stores the action associated with all transitions.
The transition table and the action table have an identical number of rows and
columns. Each table has a separate row for each state and a separate column for
each possible event or condition from the state machine. The entries in the transition table
indicate the state following the transition, sometimes called the next state.
The action table also contains entries that are appropriate for the outgoing
transition from a state and an event or condition. The entries in the action table
are actions.

The lock :

A	B	C	D	E

Finite State Machine for "BADD" combination

D / open door
other / sound alarm

OK
after 1

OK
after2

OK
after3

A

A

B

other

other

start

No key
entered

other

Fail
after1

any

Fail
after2

any

Fail
after3

any / sound alarm

In other words, the entry in the OK After 1 row and the column labeled B indicates that
from state OK After 1 a B event (pressing the B key) should make a transition into the
Fail After 2 state. The action table also contains entries that are appropriate for the outgoing
transition from a state and an event or condition. The entries in the action table
are actions.

The action table for the above contains many empty entries to represent transitions that have no
associated action.

With a 4 button sequence to unlock there are only 7 **states** that the FSM can have

nke – no key entered (aka the "start")
ok1 – ok after 1 (first button entered is part of the correct combination to unlock)
ok2 – ok after 2 (second button entered is part of the correct combination to unlock)
ok3 – ok after 3
fa1 – fail after 1 (first button entered is NOT part of the correct combination to unlock)
fa2 – fail after 2
fa3 – fail after 3

The **transition function** for the FSM is dependent upon the correct combination to unlock.

For example, if the correct unlocking combination is "BADD"
and we are in state ok1 and the user enters "A" we would transition to state ok2.
If the user was in state ok1 and entered "E" we would transition to state fa2. And so on.

The **action function** is also contingent upon the correct combination.
For example, if the user is in state ok3 and enters "D" what is the action?
If the user is in state fa2 and enters "D" what is the action ?

As discussed, tables represent functions
(a Table is really a function; the function is an input to the table constructor.
Every table will have its own mapping function.)

So for this scenario we have 2 tables: a transition table and an action table.

The transition table will be generated by a function, say t,

$$t(state, keyEntered) = newState$$

For example, from above,

$$t(ok1, A) = ok2$$

The action table contains entries that are appropriate for the outgoing
transition from a state and an event or condition. The entries in the action table
are actions. The action table for the above contains many empty entries to represent
transitions that have no associated action. For our problem, there are obviously only 2 actions

Similarly,the action table will be generated by a function, say a,

$$a(state, keyEntered) = action$$

For example, from above,

$$a(ok3, D) = ?$$

A table is an unordered group of "records", each of which has a designated field called the key field
Records within the table are identified by the value of the key field. For the operations provided by the
table see pair.h and table.h.. The table is an appropriate ADT for information retrieval systems in which
individual records must be accessed frequently (and quickly) but the entire collection of records must
never be processed sequentially.
Complete the table implementation.
use table to store the transition table.
use table to store action table.
the lock will always be the characters A , B , C , D , E.
the combination will always be a four character sequence
Must use accompanying files – with no additions/changes

DELIVERABLES:

soft: in a zipped file, called CS23_P4_yourLastName,
        submitted to blackboard CS232_P4_ANS

1. documented source code
2. user manual.
3. Programmer manual(s) (one for each class also)
4. release version executable

Due Date : 6:00am 15 May 2020