SWEN224 2015 Model Checking - Assignment 1 Total 40 points

Hand answers to all questions in one file `Ass1_Q1.txt` Build models with the same name as used in the question, add comments to the file:

```
\\ Question1 VMhit
```

Put your answers in the same order they appear in the assignment.

1. [**5 points**] Build the processes

```
Ping = one->three-> STOP.
Pong = two->STOP.
Opps = Ping||Pong.
```

   Change `Ping` and `Pong` by adding events `x` and `y` so that

```
Good = simp(abs((PingX||PongX)\{x,y})).
```

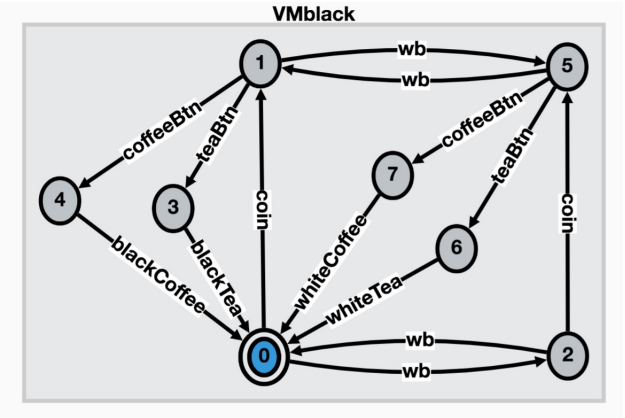
   is as shown above.

2. [**10 points**] Model the following non-terminating vending machines with slots to insert coins and pay for your drink, buttons to select your drink. The machine only vends your drink after you have paid and pushed a button.

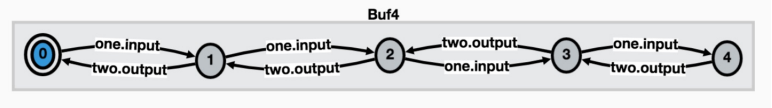
   (a) `VMhit` Requires one coin, a coffee button that only works after a coin has been inserted and only then allows you to take your coffee. Plus if you hit the vending machine after you have inserted a coin and before you take your coffee you loose your money.

   (b) `VMtc` has a tea and a coffee button. This vending machine charges one coin for tea but two coins for coffee.

   (c) `VMblack` is a vending machine that offers tea and coffee. But this vending machine can be in one of two states: in the first state it will dispense black drinks and in the second it will dispense white drinks. The vending machine can be toggled between the two states by pressing a button, event `wb`, To change state this button has to be pushed directly before or after inserting the `coin`. See the figure below:



3. [**5 points**] Build a two place buffer make two copies of the two place buffer and compose them in parallel gluing the output from one to the input of the other there by building a four place buffer.



4. [**10 points**] In this question you need to define a radio button `RBt` process that controls a lift. The radio button has three states:

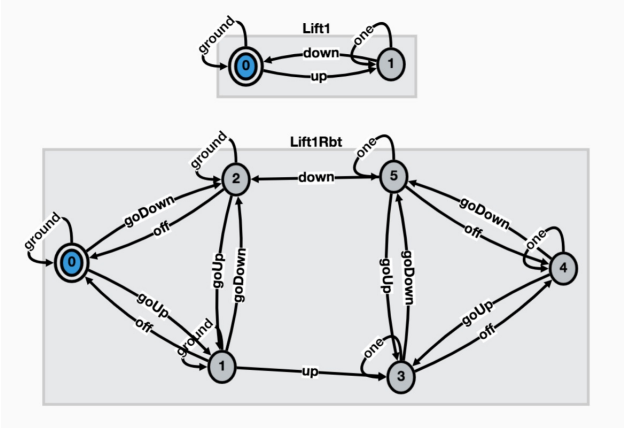
   (a) both buttons off

   (b) only the up button on and

   (c) only the down button on.

   A person can change the state of the radio button by executing one of the three events `off`, `goUp` and `goDown`. The radio button moves the lift with the events `up` and `down`.

   The Lift and radio button together behave like the process `Lift1RBt = Lift1 || Rbt` shown below.



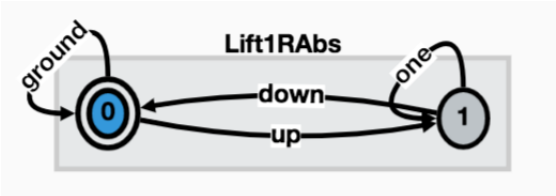
   (a) Build `Lift1` that moves between two floors `one` and `ground` see above:

   (b) Now define the `RBt` so that when it is run in parallel with `Lift1` together they will behave like `Lift1RBt = Lift1 || Rbt`

   (c) As a sanity check hide the person - button events `off`, `goUp` and `goDown` with term:

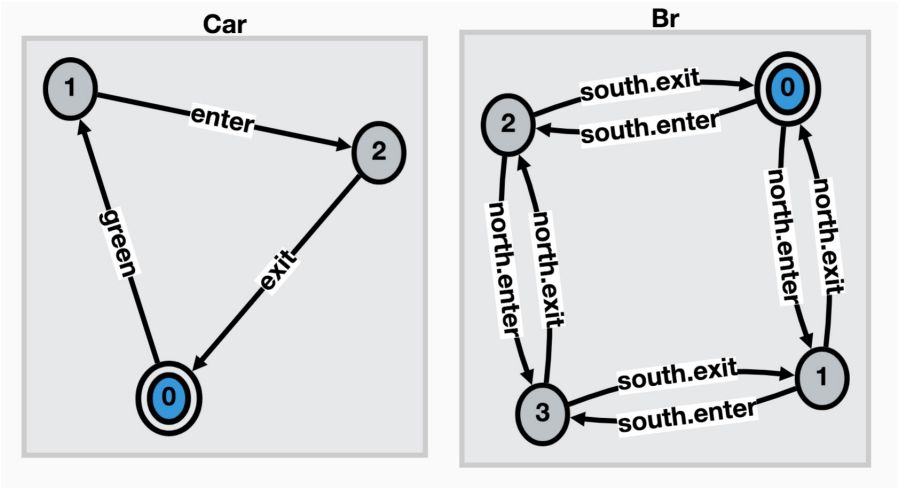
```
Lift1RAbs = simp(abs((Lift1||RBtn)\{off,goUp,goDown}))
```

   and check it has the behaviour shown below.



5. [**10 points**] `Car`s can cross a one way bridge from the `north` or the `south`. **Define a bridge traffic light `BridgeTL`.** the process `BridgeTL` has two states, in one the `south.green` can be seen, but not the `north.green` and in the other the `north.green` can be seen, but not the `south.green`.

   To make the traffic light change state the event `north.tored` ends in the state where the `south.green` event can be seen. The event `south.tored` ends in the state where the `north.green` event can be seen.



Two cars, `north:Car` and `south:Car` can be controlled by the traffic light so that only one is on the bridge at a time. Check this by constructing :

```
Br = simp(abs((north:Car || south:Car || BridgeTL)
      \{north.green,north.tored,south.green,south.tored}))
```

and checking that it behaves as in the figure above.