Problem Set 3
Safe and Secure Software (WS 11/12)

Bauhaus-University Weimar, Chair of Media Security
Prof. Dr. Stefan Lucks, Christian Forler
Url: `http://www.uni-weimar.de/cms/medien/mediensicherheit`

Problem 1: Testgen
Read and understand the `testgen` implementation from here.

Problem 2: RGB Test Driver (2 Points)
Use `testgen` to find new bugs in the RGB implementation of your fellow students. Therefore, you have to write a test driver for the RGB package. Further Information including a documentation can be found at this link: http://www.free-software-consulting.com/projects/tg/.

```
package RGB is
   type Color is private;
   subtype Intensity is Integer range 0..255;

   function To_Color(Red    : Intensity;
                     Green  : Intensity;
                     Blue   : Intensity)
                    return Color;

   -- Saturation arithmetics
   function "+"(Left : Color; Right : Color)  return Color;
   function "-"(Left : Color; Right : Color)  return Color;
   function "*"(Left : Color; Right : Color)  return Color;

   -- print the Intensity of each color as hex values.
   procedure Put(Item : in Color);

private
   type RGB is (Red, Green, Blue);
   type Color is array (RGB) of Intensity;
end RGB;
```

Problem 3: Coffee Machine (2 Points)
Implement the following specification and write a test `testgen` driver for it.

```
package Coffee Machine is
   -- Simulation of a coin-driven coffee machine
   -- User: - One slot to insert coins (only, 10 or, 20 cents)
   --       - One button to press (''money back'')
   -- Machine: one slot to drop coins, the coffee output
   -- Given 30 cents or more, the coffee is produced immediately
   -- (Note that Overspending is Possible)

   type State is private;
   type Action is(Ten_Cent, Twenty_Cent, Button);
   type Reaction is(Nothing, Drop_All_Coins, Coffee);

   procedure Initialize( X : out State);
   procedure X(S      : in out State;
               Act    : in Action;
               React  : out Reaction);

private
   type State is range 0..2;
end Coffee_Machine;
```

Mini-Project 4: Graph (4 Points)
Implement the following specifications.

```
generic
   type Vertex_Type is private;
package Generic_Graph is
   type Graph_Type is tagged limited private;
   subtype Edge_Weight is Natural;
   type Vertex_Array is array (Positive range <>) of Vertex_Type;

   -- turns the graph into an empty graph without vertices and edges
   procedure Clear (Graph : in out Graph_Type);


   -- inserts new Vertex; raises Constraint_Error if Vertex is already there
   -- the new Vertex is unmarked
   procedure Add_Vertex (Graph  : in out Graph_Type; Vertex : in Vertex_Type);


   -- inserts an Edge into the graph, raises Constraint error if Head or
   -- Tail aren't already in the graph; overwrites an Edge if it already
   -- exists (i.e., changes the weight)
   procedure Add_Edge (Graph: in out Graph_Type;
                       Head, Tail: in Vertex_Type;
                       Weight: in Edge_Weight);


   -- returns the weight of an edge; Natural'Last if the Edge doesn't exist
   function Weight_Of (Graph: Graph_Type;
                       Head, Tail: Vertex_Type) return Edge_Weight;
   -- all nodes K with an edge from Vertex to K
   function Successors (Graph  : Graph_Type;
                        Vertex : Vertex_Type) return Vertex_Array;

   -- all nodes K with an edge from K to Vertex
   function Predecessors (Graph  : Graph_Type;
                          Vertex : Vertex_Type) return Vertex_Array;

   -- return all vertices in the Graph
   function All_Vertices(Graph : Graph_Type) return Vertex_Array;


private
   -- implementation dependent ...
end Generic_Graph;




with Generic_Graph;

generic
   type Vertex_Type is private;
   type Vertex_Mark is (<>);
package Mark_Graph is

   package Graphs is new Generic_Graph(Vertex_Type);

   type Graph_Type is new Graphs.Graph_Type with private;

   procedure Set_Mark(Graph  : in out Graph_Type;
                      Vertex : in     Vertex_Type;
                      To     : in     Vertex_Mark);

   function Get_Mark (Graph  : Graph_Type;
                      Vertex : Vertex_Type) return Vertex_Mark;

private
    -- implementation dependent ..
end Mark_Graph;
```

Mini-Project 5: Graph Test Driver (4 Points)
Write `testgen` test drivers for the above mini-project.