```
In[47]:= (* Coin flip, problem 3 *)

In[48]:= (* Assumption: Coin radius is 1, height is 1 *)

In[49]:= (* Time to go back to initial height *)

In[50]:= timeFunction[v_] := 2 v / 9.8
         timeFunction[4.5]

Out[51]= 0.918367

In[52]:= (* Angular velocity is in radians *)
         (* converts angular velocity to degrees per second *)
         angularToDegrees[a_] := N[a * (180 / Pi)]

         (* Function to find where the coin is roated once it falls back into initial height *)
         finalDegrees[timetofunction_, degrees_] := timetofunction * degrees

         (* Reduces the finalDegrees into a value from 0 to 360 degrees *)
         reducetobounds[finaldegrees_] := finaldegrees - (360 * Floor[finaldegrees / 360])

         (* Takes a reduced degree and finds if it will land heads or tails. 1 = heads,
         0 = tails *)
         headsortails[reducedDegree_] := If[(reducedDegree > 270 || reducedDegree < 90), 1, 0]

In[56]:= (* This is where I am going to introduce error *)

In[57]:= (* To land on the side, the coin needs to rotate exactly 90 or 270 degrees *)
         (* However this will never happen as the
          precision of my calculation always has a decimal *)
         (* Instead of rounding to the nearest n-th degree, I am going to take a ratio *)
         (* If reducedDegree/90 or reducedDegree/270 is between .999 and 1.001,
         it lands on its side *)
         (* 1 = lands on side, 0 = no *)
```

```
In[58]:= side[reducedDegree_] := If[((reducedDegree / 90 ≥ .999 && reducedDegree / 90 ≤ 1.001) ||
             (reducedDegree / 270 ≥ .999 && reducedDegree / 270 ≤ 1.001)), 1, 0];

         (* Implement a function to use all the above functions. Takes in
           a velocity and an angular momentum. 1 = heads, 0 = tails, 2 = side *)
         coinFlip[v_, w_] := (
           time = timeFunction[v];
               degrees = angularToDegrees[w];
               totalDegrees = finalDegrees[time, degrees];
               actualDegree = reducetobounds[totalDegrees];
               If[side[actualDegree] == 1, Return[2], Return[headsortails[actualDegree]]])

         headsListv = {};
         headsListw = {};
         tailsListv = {};
         tailsListw = {};
         sideListv = {};
         sideListw = {};

In[66]:=

In[67]:= a = 0;
         b = 0;

In[69]:= While[a < 10,
           While[b < 10,
            result = coinFlip[a, b];
            If[result == 1, (AppendTo[headsListv, a]; AppendTo[headsListw, b])];
            b = b + 1
           ];
           b = 0;
           a = a + 1
          ];
```