```
In[507]:= (* Coin flip, problem 3 *)

In[508]:= (* Assumption: Coin radius is 1, height is 1 *)

In[509]:= (* Time to go back to initial height *)

In[510]:= timeFunction[v_] := 2 v / 9.8
          timeFunction[4.5]

Out[511]= 0.918367

In[512]:= (* Angular velocity is in radians *)
          (* converts angular velocity to degrees per second *)
          angularToDegrees[a_] := N[a * (180 / Pi)]

In[513]:= (* 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[516]:= (* This is where I am going to introduce error *)

In[517]:= (* 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 .99 and 1.01,
          it lands on its side *)
          (* 1 = lands on side, 0 = no *)
```

In[518]:=
```
side[reducedDegree_] := If[((reducedDegree / 90 ≥ .99 && reducedDegree / 90 ≤ 1.01) ||
    (reducedDegree / 270 ≥ .99 && reducedDegree / 270 ≤ 1.01)), 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[526]:=

In[527]:=
```
a = 0;
b = 0;
```

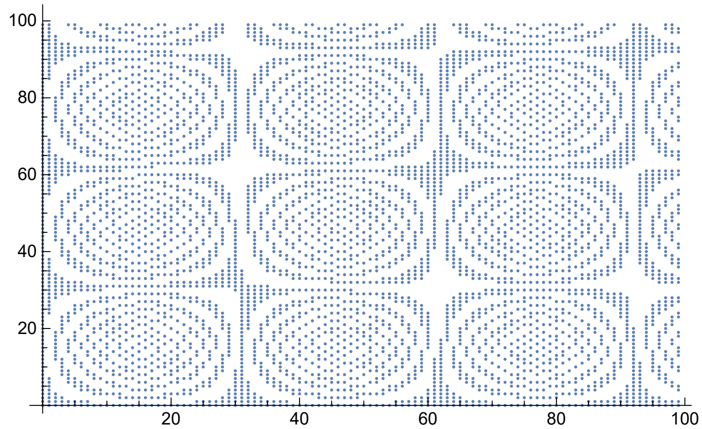
In[529]:=
```
While[a < 100,
  While[b < 100,
   result = coinFlip[a, b];
   If[result == 1, (AppendTo[headsListv, a]; AppendTo[headsListw, b])];
   If[result == 0, (AppendTo[tailsListv, a]; AppendTo[tailsListw, b])];
   If[result == 2, (AppendTo[sideListv, a]; AppendTo[sideListw, b])];
   b = b + 1
  ];
  b = 0;
  a = a + 1
 ];
```

In[530]:=
```
headsTable = Transpose[{headsListv, headsListw}];
tailsTable = Transpose[{tailsListv, tailsListw}];
sidesTable = Transpose[{sideListv, sideListw}];
```

In[533]:=

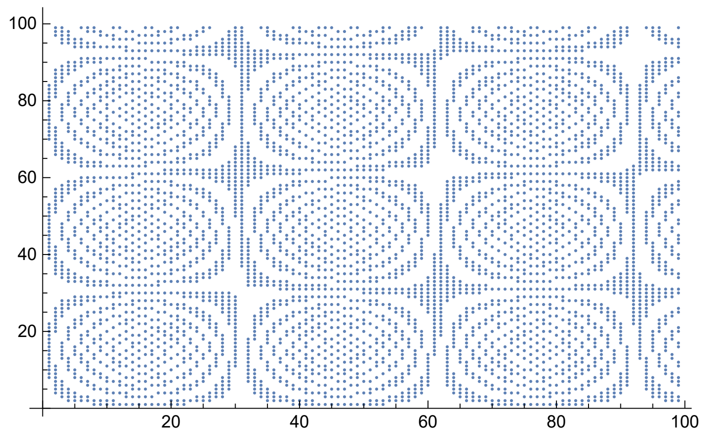
In[534]:= `ListPlot[headsTable]`

Out[534]=



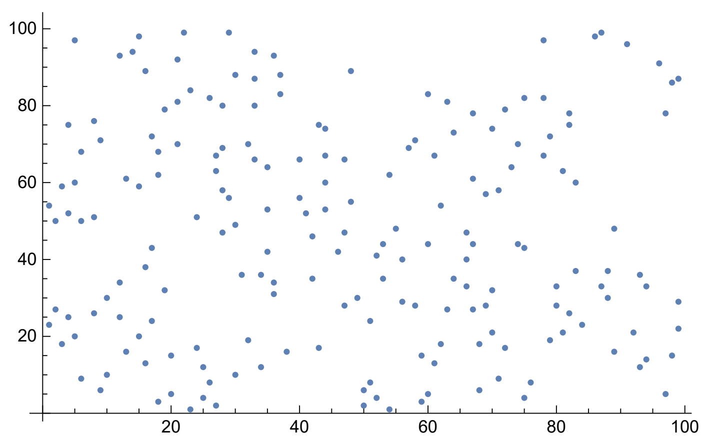
In[535]:= `ListPlot[tailsTable]`

Out[535]=



In[536]:= `ListPlot[sidesTable]`

Out[536]=



In[537]:=

```
(* Results for maximum v and w values: *)
(* 50: .502 heads, .4796 tails, .0184 side *)
(* 100: .4964 heads, .4842 tails, .0194 side *)
(* 150: .4934 heads, .4856 tails, .020 side *)
(* Conclusion: As v and w increase, the ratio of heads and tails approaches 50/50 *)
totalLength = Length[headsListv] + Length[tailsListv] + Length[sideListv]
```

Out[537]= 10 000

In[538]:=

```
headsratio = N[Length[headsListv] / totalLength]
tailsratio = N[Length[tailsListv] / totalLength]
sideratio = N[Length[sideListv] / totalLength]
```

Out[538]= 0.4964

Out[539]= 0.4842

Out[540]= 0.0194

In[541]:=

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

In[547]:=

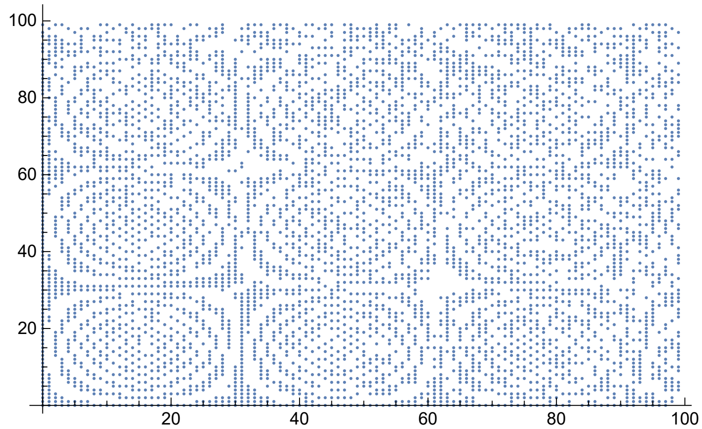
```
a = 0;
b = 0;
While[a < 100,
  While[b < 100,
   vError = RandomReal[{-.1, .1}] + a;
   wError = RandomReal[{-.1, .1}] + b;
   result = coinFlip[vError, wError];
   If[result == 1, (AppendTo[headsListv, a]; AppendTo[headsListw, b])];
   If[result == 0, (AppendTo[tailsListv, a]; AppendTo[tailsListw, b])];
   If[result == 2, (AppendTo[sideListv, a]; AppendTo[sideListw, b])];
   b = b + 1
  ];
  b = 0;
  a = a + 1
 ];
```

In[550]:=

```
headsTable = Transpose[{headsListv, headsListw}];
tailsTable = Transpose[{tailsListv, tailsListw}];
sidesTable = Transpose[{sideListv, sideListw}];
```

In[553]:=
```
ListPlot[headsTable]
```

Out[553]=



In[554]:=

In[557]:=
```
totalLength = Length[headsListv] + Length[tailsListv] + Length[sideListv]
```

Out[557]= 10 000

In[558]:=
```
headsratio = N[Length[headsListv] / totalLength]
tailsratio = N[Length[tailsListv] / totalLength]
sideratio = N[Length[sideListv] / totalLength]
```

Out[558]= 0.4909

Out[559]= 0.4888

Out[560]= 0.0203

In[555]:=

In[556]:=