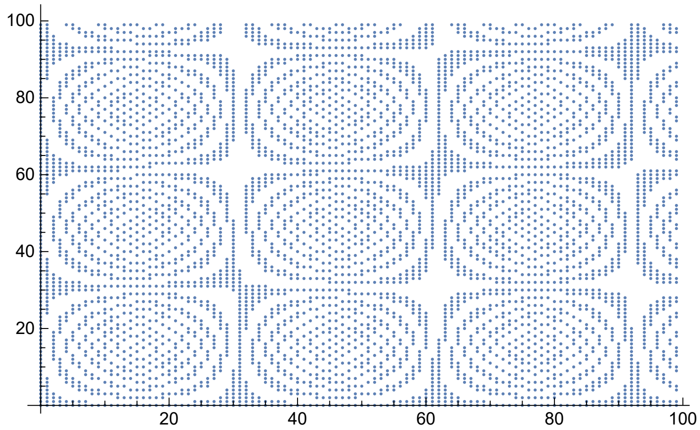
```
In[335]:= (* Coin flip, problem 3 *)

In[336]:= (* Assumption: Coin radius is 1, height is 1 *)

In[337]:= (* Time to go back to initial height *)

In[338]:= timeFunction[v_] := 2 v / 9.8
          timeFunction[4.5]

Out[339]= 0.918367

In[340]:= (* Angular velocity is in radians *)
          (* converts angular velocity to degrees per second *)
          angularToDegrees[a_] := N[a * (180 / Pi)]

In[341]:= (* 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[344]:= (* This is where I am going to introduce error *)

          (* 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[346]:= 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[354]:=

In[355]:= a = 0;
          b = 0;

In[357]:= 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[358]:= headsTable = Transpose[{headsListv, headsListw}];
          tailsTable = Transpose[{tailsListv, tailsListw}];
          sidesTable = Transpose[{sideListv, sideListw}];

In[361]:=
```

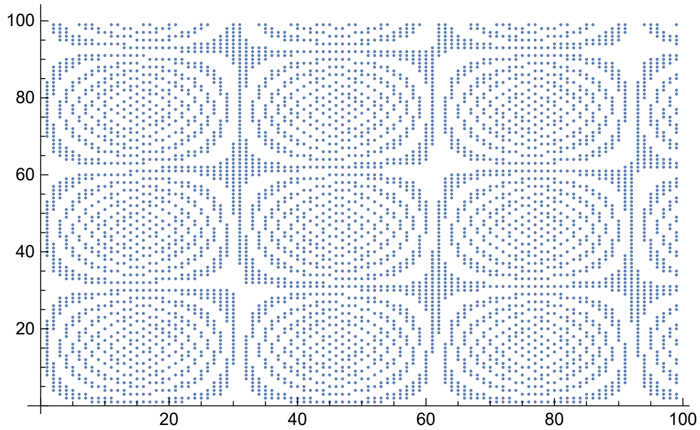
In[362]:=
```
ListPlot[headsTable]
```

Out[362]=



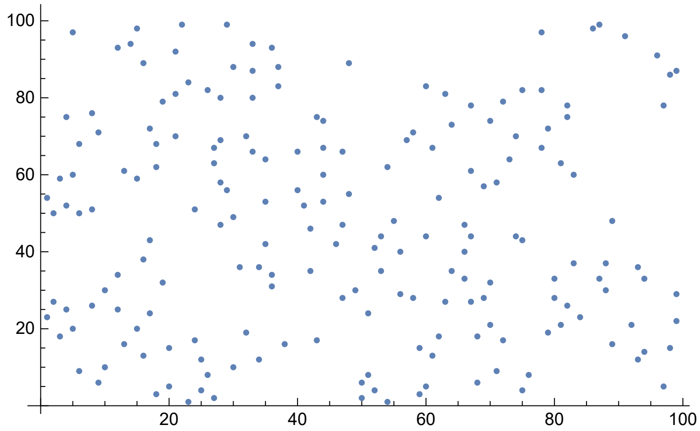
In[363]:=
```
ListPlot[tailsTable]
```

Out[363]=



In[364]:=
```
ListPlot[sidesTable]
```

Out[364]=


100
80
60
40
20
20
40
60
80
100