Galton Board

Application of dynamic memory allocation

outline

- Introduction, see https://www.youtube.com/shorts/Kq7e6cj2nDw
- Apply dynamic spaces to represent a 2-dimensional array where number of columns differ from row to row.
- Implementation
- Read: Section 7.5, Arrays and Vectors of Pointers

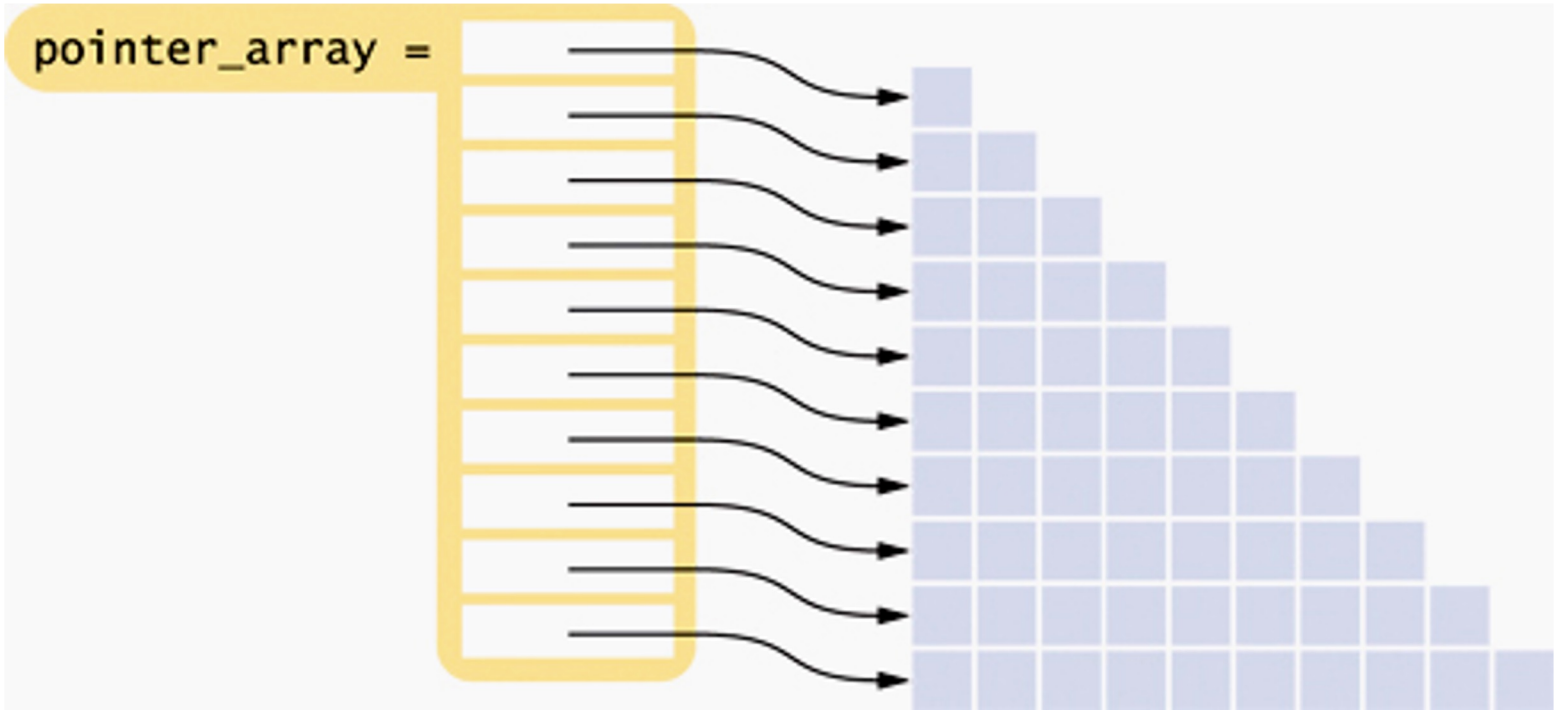
Arrays and Vectors of Pointers

When you have a sequence of pointers,
you can place them into an array or vector.

An array and a vector of ten **`int*`** pointers are defined as

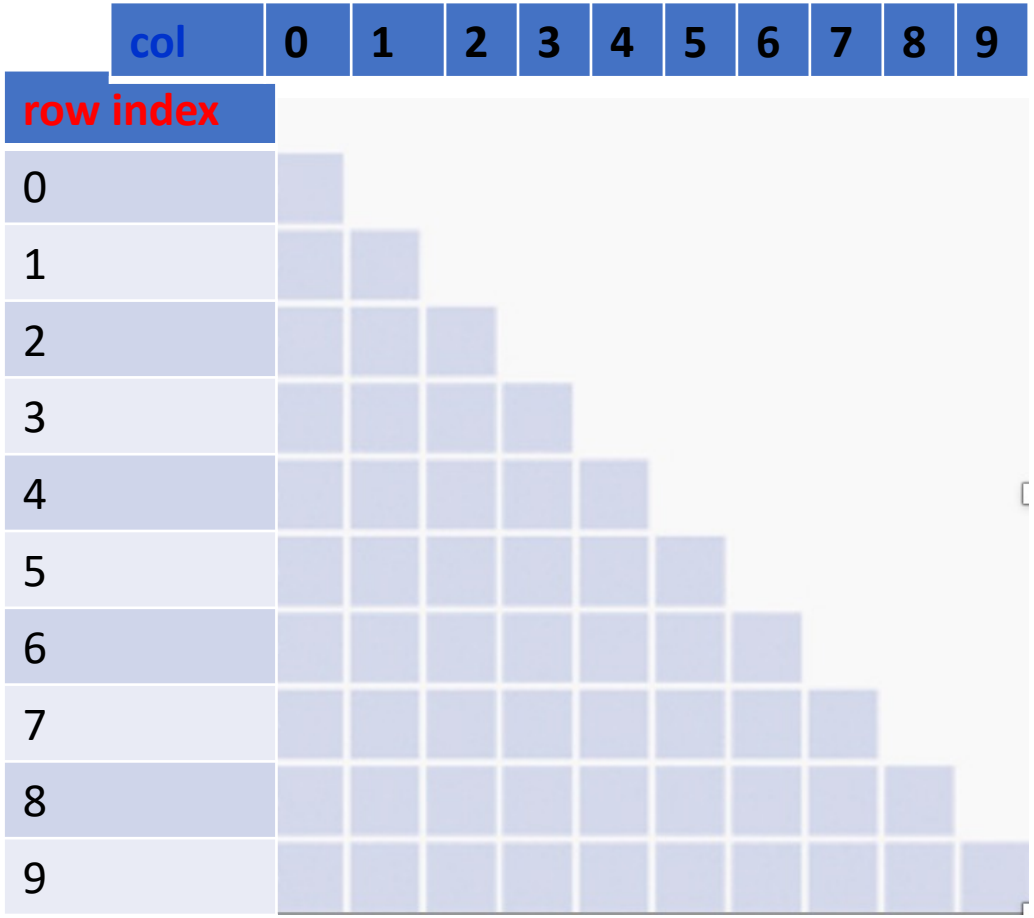
```
int* pointer_array[10];

vector<int*> pointer_vector(10);
```

Arrays and Vectors of Pointers – A Triangular Array



In this array, each row is a different length. It would be inefficient to use a two-dimensional array, because almost half of the elements would be wasted

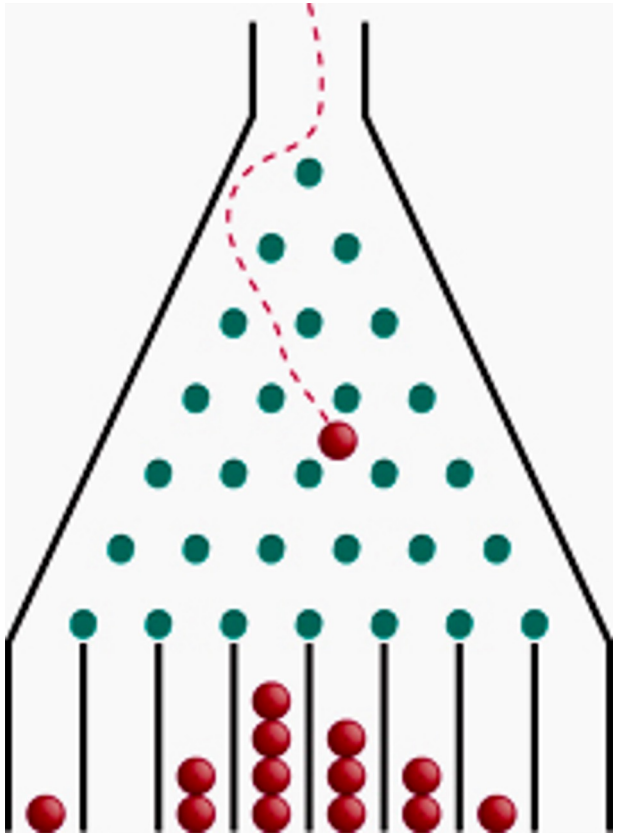
A Triangular Array



In this array, each row is a different length. It would be inefficient to use a two-dimensional array, because almost half of the elements would be wasted

Program Example: A Galton Board

A Galton board consists of a pyramidal arrangement of pegs and a row of bins at the bottom.



Balls are dropped onto the top peg and travel toward the bins.

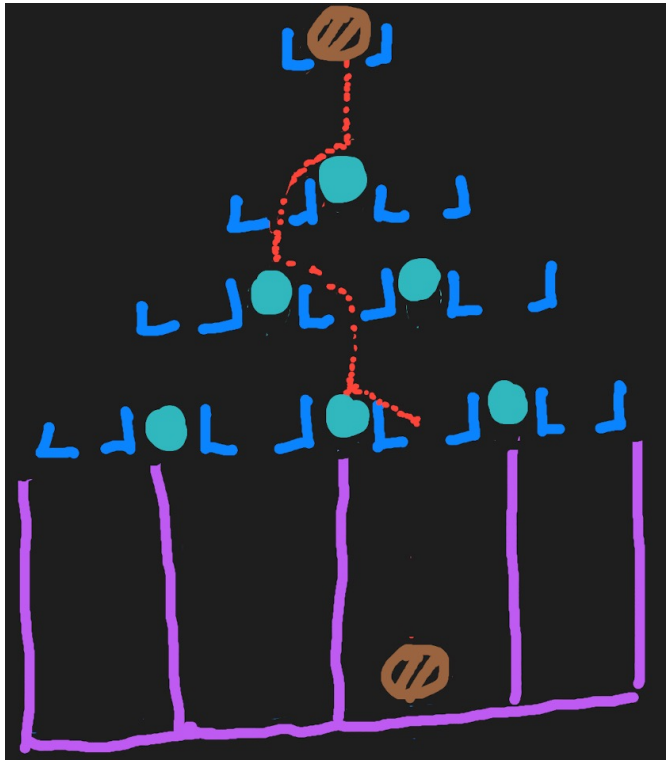
At each peg, there is a 50 percent chance of moving left or right.

The ball counts in the bins approximate a bell-curve distribution.

https://www.youtube.com/watch?v=6YDHBFVIvIs

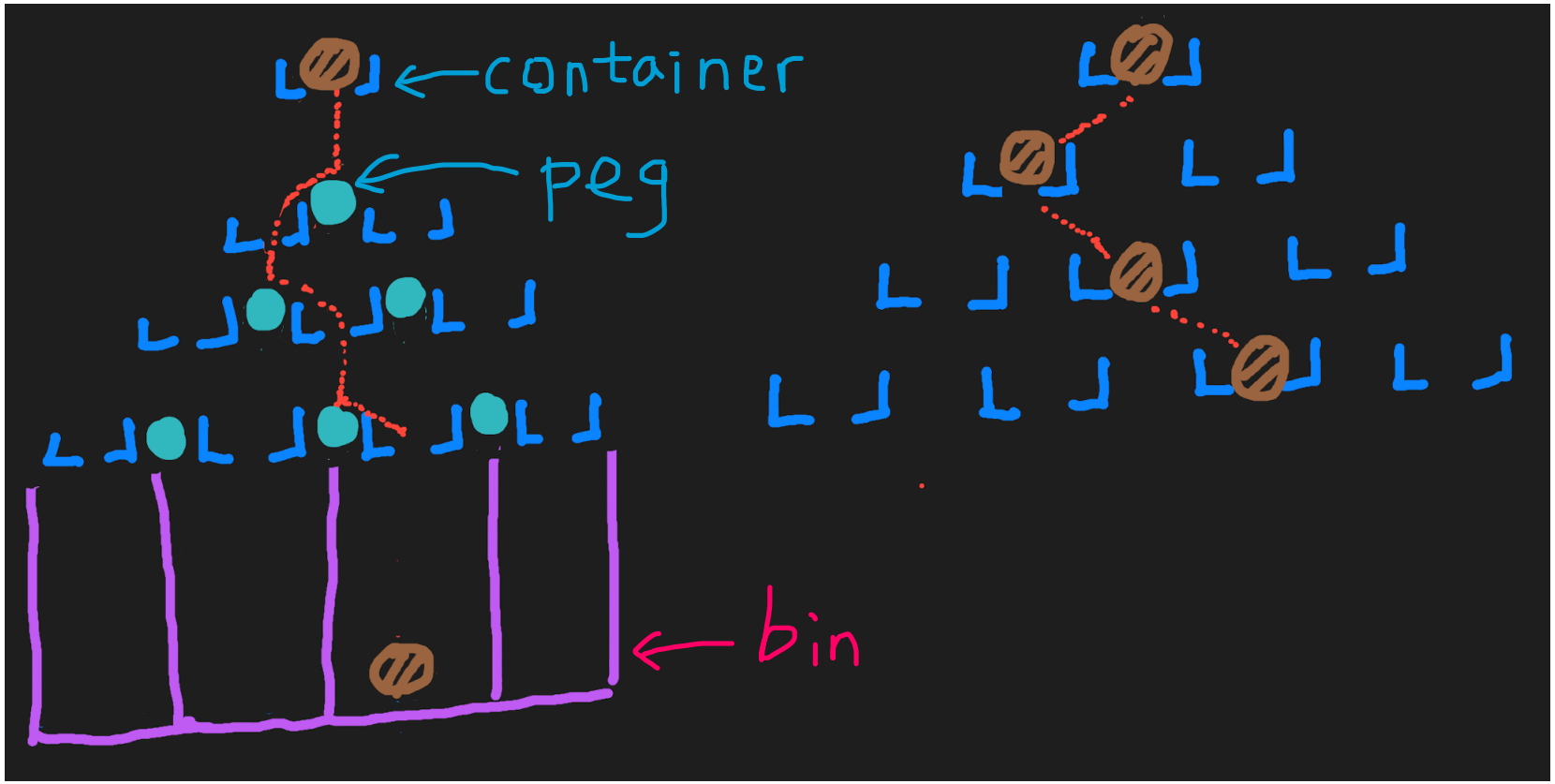
Conversion

- Image containers are before, between, and after pegs. Once a ball is bounded to the left, the ball passes the container left to the peg, otherwise, the ball passes the container right to the peg.

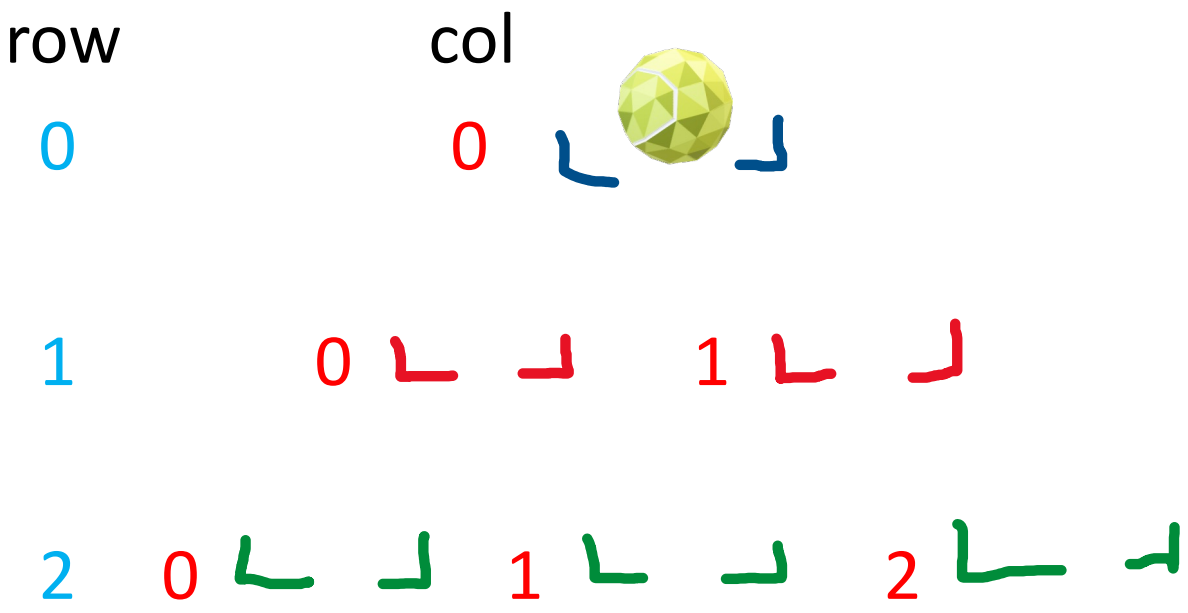


Conversion: II

- Each container records the number of balls passing it.
- How to represent the containers?

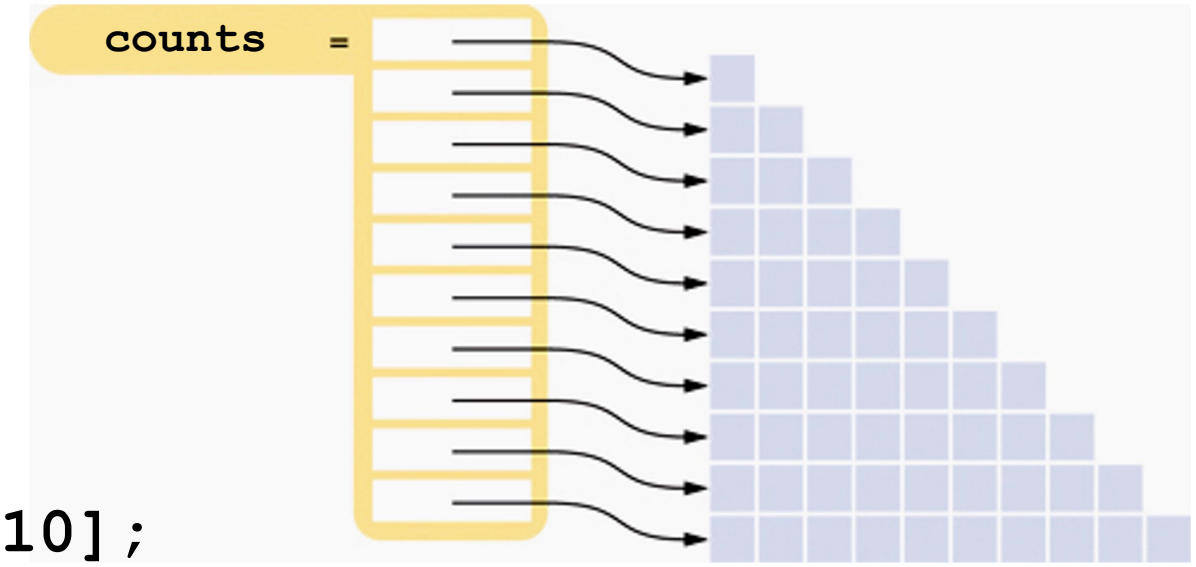


Galton board: throw a ball



A Galton Board Simulation

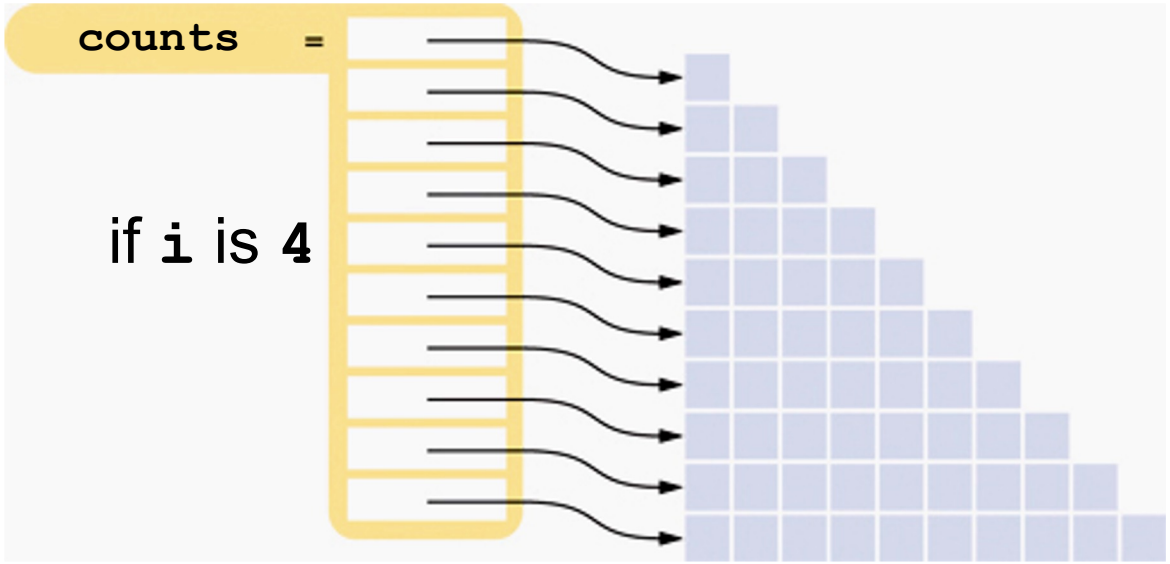
We will simulate a board with ten rows of pegs.
Each row requires an array of counters.
The following statements initialize the triangular array:




```
int* counts[10];
for (int i = 0; i < 10; i++)
{
   counts[i] = new int[i + 1];
}
```

A Galton Board Simulation: Printing Rows

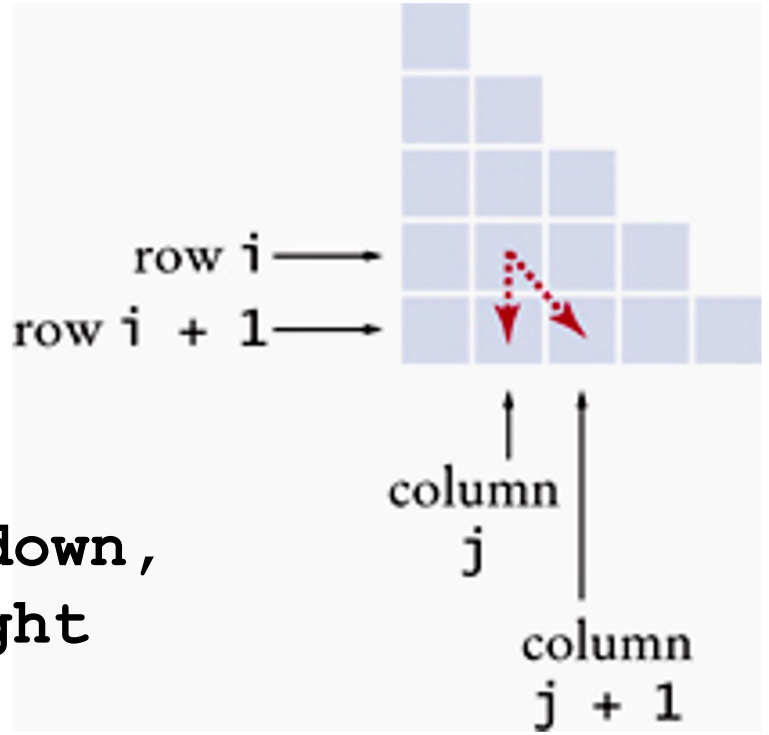
We will need to print each row:




```
// print all elements in the ith row
for (int j = 0; j <= i; j++)
{
    cout << setw(4) << counts[i][j];
}
cout << endl;
```

A Galton Board Simulation: Ball Bouncing on Pegs

We will simulate a ball bouncing through the pegs:




```
int r = rand() % 2;
// If r is even, move down,
// otherwise to the right
if (r == 1)
{
   j++;
}
counts[i][j]++;
```

A Galton Board Simulation: Complete Code Part 1

```
#include <iostream>
#include <iomanip>
#include <cstdlib>
#include <ctime>
using namespace std;
int main()
{
   srand(time(0));
   int* counts[10];

   // Allocate the rows
   for (int i = 0; i < 10; i++)
   {
      counts[i] = new int[i + 1];
      for (int j = 0; j <= 1; j++)
      {
         counts[i][j] = 0;
      }
   }
```

A Galton Board Simulation: Complete Code Part 2

```
const int RUNS = 1000;
// Simulate 1,000 balls
for (int run = 0; run < RUNS; run++)
{
   // Add a ball to the top
   counts[0][0]++;
   // Have the ball run to the bottom
   int j = 0;
   for (int i = 1; i < 10; i++)
   {
      int r = rand() % 2;
      // If r is even, move down,
      // otherwise to the right
      if (r == 1)
      {
         j++;
      }
      counts[i][j]++;
   }
}
```

A Galton Board Simulation: Complete Code Part 3

```
   // Print all counts
   for (int i = 0; i < 10; i++)
   {
      for (int j = 0; j <= i; j++)
      {
         cout << setw(4) << counts[i][j];
      }
      cout << endl;
   }

   // Deallocate the rows
   for (int i = 0; i < 10; i++)
   {
      delete[] counts[i];
   }

   return 0;
}
```

A Galton Board Simulation: Results

This is the output from a run of the program, with each number being a count of the balls that hit that peg in the triangle.

Note the bell-curve distribution of balls on the "bottom line":

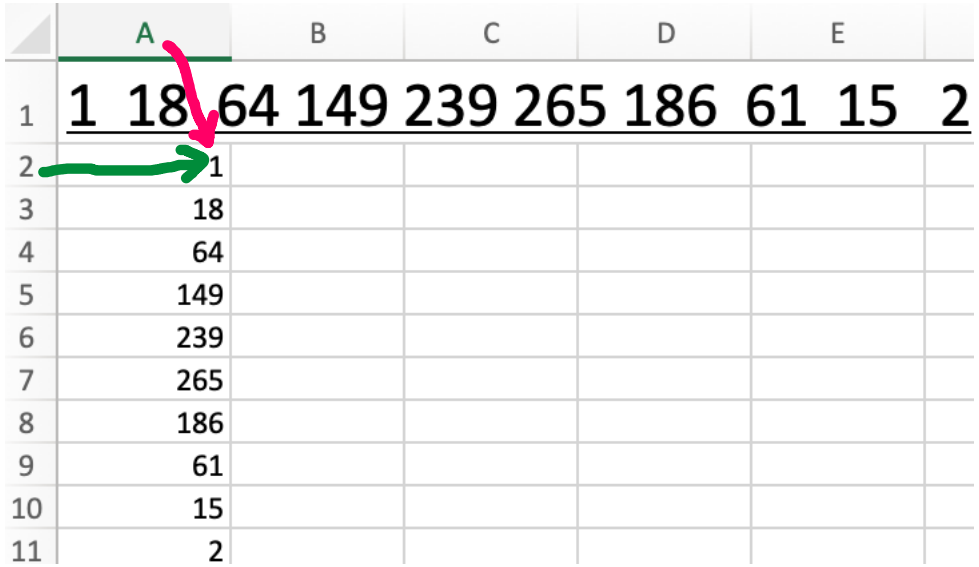
```
1000
 480 520
 241 500 259
 124 345 411 120
  68 232 365 271  64
  32 164 283 329 161  31
  16  88 229 303 254  88  22
   9  47 147 277 273 190  44  13
   5  24 103 203 288 228 113  33   3
   1  18  64 149 239 265 186  61  15   2
```

Draw a column chart in excel (optional)

- Copy the last line of the previous slides (or your output) in the first cell of excel for temporary storage.

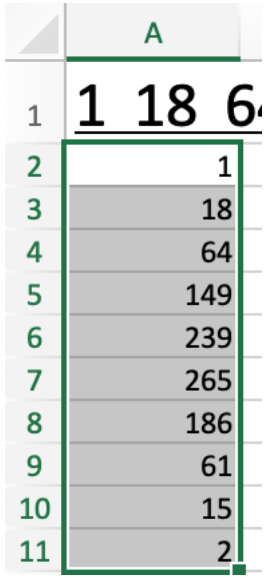
1 18 64 149 239 265 186 61 15 2

- Then type in the corresponding values in cells A2 – A11 in excel.



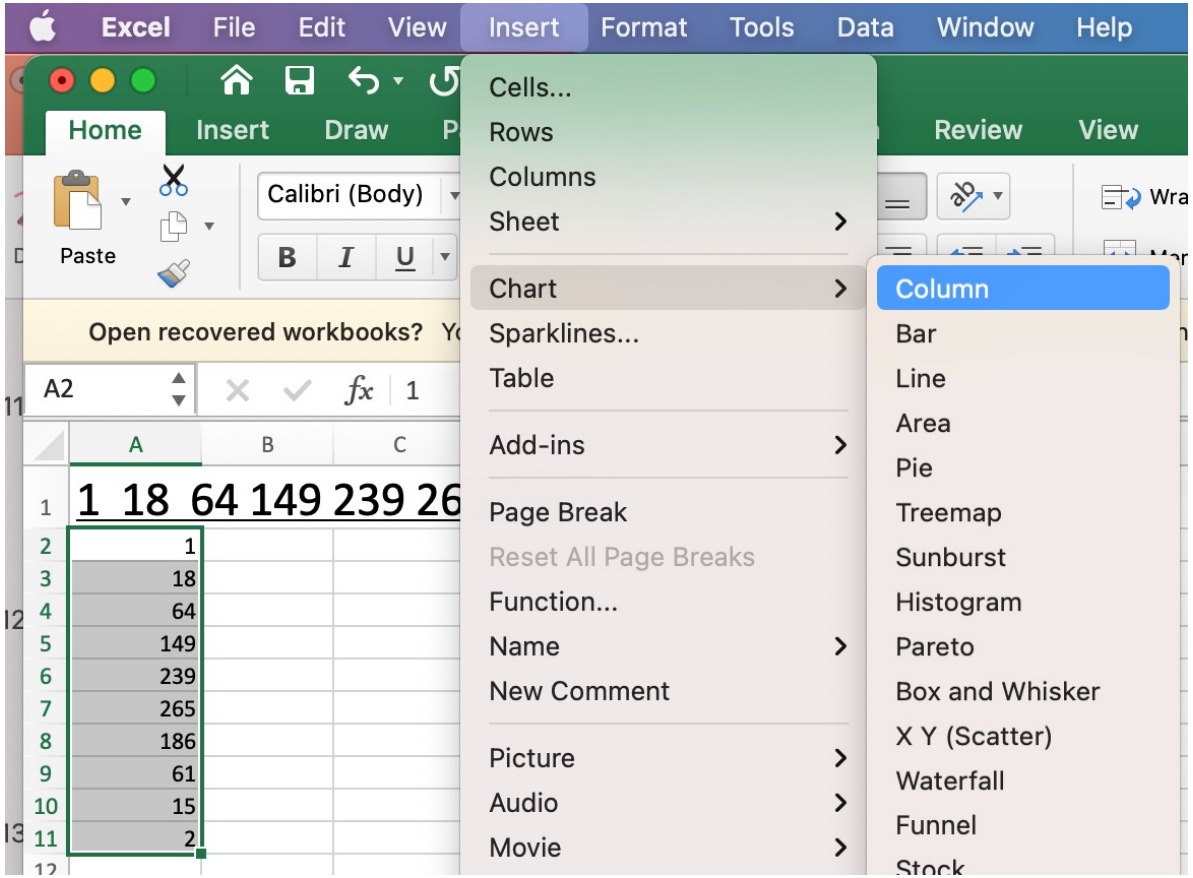
Draw a column chart in excel (optional): II

- Highlight cells with data, in this example, from A2 to A11.
- Click A2, then press Shift key, at the same type, click A11. Then all cells from A2 to A11 are selected.



Draw a column chart in excel (optional): III

- Then in excel menu, choose Insert->chart->column.



Draw a column chart in excel (optional): IV

- A chart graph is drawn.