**Simulation Lab 05**

**random dropping:**

**The main algorithm**

We can have a list for the bins with the number of balls inside the bins as the values for each place in the list.

With each ball that goes in the bin, we increase the number inside of it.

So

I think that using a list is the best option for the **data structure** side of the simulation

The **input** of the code will be the size of the list which is equal to the number of the bins and balls

The **output** of the simulation is a list. Each index represents a bin and the number associated with it is the number of balls inside of it.

We **assume** that the length of the list fixed

We randomly choose a bin which is technically an index of the list

For the data structure part, we could also use a **dictionary** like this:  
{“number of the bin”:number of balls in the bin}

**Random load balancing with d = 2 and d=4**

In this part we can again use a **list** or a **dictionary** as the **data structure** in which we store the bins and the balls inside them.

We randomly choose 2/4 of the indexes of the list then we do a comparison to find the least occupied index in the list (place in the list)

The number of bins which is the length of the list is fixed

The input of the simulation is d and N (length of the list/ number of bins and balls)

In terms of occupancy, random load balancing with d = 4 is a better option than the others in term of average occupancy cause it’s taking more bin in account while choosing the bin to fill with balls compared to d =2. When it comes to random approach there’s a high possibility of many bins being empty and many bins having more than one ball inside of them so in term of maximum occupancy, the random approach is not a good option since we want all bins to be filled with one ball, but this output has a low probability to happen.