1.

The first person has a 100% chance of a unique number (of course)

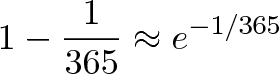
The second has a (1 – 1/365) chance (all but 1 number from the 365)

The third has a (1 – 2/365) chance (all but 2 numbers)

The nth has a (1 – (n-1)/365) chance (all but n-1 numbers)

P(different) = 1\*(1-)\*(1-)\*……\*(1- )

Since \displaystyle{e^x  \approx 1 + x} ,

So, 

P(different) ≈e-(1+2+……+n-1)/365≈e-(n^2/(2\*365)

P(match) = 1 – P(different) ≈1- e-(n^2/(2\*365)

P(match) ≈1- e-(n^2/(2\*365)

\displaystyle{n \approx \sqrt{-2ln(1-m)} \cdot \sqrt{T}} m=P(match), T= days

Program:

Import Java.lang.Math;

Public int calculate(float m, int T){

double temp = (Math.log(1-m)/Math.log(10))\*(-2)\*T;

int n = Math.sqrt(temp);

return n;

}

2. Design an algorithm for the hash function you would use to "hash" by birthdays. Describe the input, the table size, and your function. How will you handle collisions? Will the size of the data affect the way you handle collisions?

Algorithm:

Input: l bits numbers

Table size: 2l/2

Firstly, choosing a random initial value x0 ∈ {0, 1} l+1 and then, for i = 1, . . . 2l/2, computing xi : H(xi−1) and x2i : H(H(x2(i−1))).

In each step the values xi and x2i are compared. If they are equal, there is a collision somewhere in the sequence x0, x1, . . . ,x2i−1 and the algorithm runs a sub-routine to find it.

Using opening hash by chaining (link list) to handle collision. The size of data will not affect my way handling the collisions.