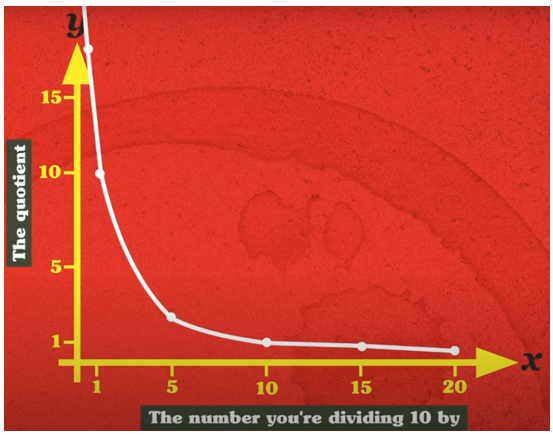
First of all dividing a positive number by zero produces a positive infinity (INF or Infinity)

Dividing by 0 is usually undefined and it's a different topic on its own, however the idea of infinity comes as the smaller the divisor (approaching zero) >> the bigger the quotient (if the dividend remains the same)

An illustration of the argument above:



Hence, Our compiler just prints out 1.#INF for a positive infinity (cause of dividing by 0). There is 1 because the C runtime library arbitrarily decided that such exceptional values (infinity) have one digit before the decimal (So, 1) according to Raymond Chen (Microsoft Blog). Then, we have the decimal point (.) due to the variable type being float. The "#INF" is there as an exceptional set value as explained earlier.

Based on our program the two unusual values we have are "1.#IO" and "1.#J". Both of these are rounded up "1.#INF". It is because we have indicated that our float values would only have either 2 or 3 digits after the decimal point. The first space is occupied by #. To obtain #IO we round up #INF, we get rid of the F and add an increment to N => O. Similarly, to obtain #J from #INF we get rid of the N and add an increment to I => J.

```
printf("\t10/0 = %0.2f\n", avg(10,0));
```

```
printf("The 4 averages are %0.3f, %0.3f, %0.3f, %0.3f",avg(55,10),avg(9,3),avg(0,5),avg(10,0));
```