Exercise #4.2 Loop Control

Loop control is a quite useful flow control when we want to the program to execute certain lines of code repeatedly. For example, if we want to increment a variable by one for one hundred times, we don't have to write "i=i+1" for one hundred time but use while or for loop to save our time. The difficult part is to conclude which lines of code are executed repeatedly and what information is not useful for next round of loop.

Problem descriptions:

Write a Java program *PandemicSpread* to simulate how *fast* a pandemic spread with the following input:

- init : the initial number of infected persons on day 1
- numInfect : the average number of healthy people that one infected person newly infects per day
- population : the total number of people in the area

Return the day on which the *entire* population will be infected.

Key information

New cases = init * numInfect

Infection spreads from day 2

Solution Design

The best way to conclude a correct and general patten for loop starts from simulations.

With the given examples

Example 1: init =1 numInfect=2 population=10

day	existed cases	new cases	Total cases
2	1	1*2=2	1+2=3
3	3	3*2=6	3+6=9
4	9	9*2=18	9+18=27>10

Thus, it takes 4 days to infect all population

Example 2: init =5 numInfect=3 population=1000

Day	existed cases	new cases	Total cases

2	5	5*3=15	5+15=20
3	20	20*3=60	20+60=80
4	80	80*3=240	80+240=320
5	320	320*3=960	1280

Thus, it takes 5 days to infect all population

Patten summarization:

For any existed cases i on the first day and infection rate r:

Day	existed cases	New Cases	Total cases
2	i	i*r	i+i*r
3	i+i*r	(i+i*r)*r	(i+i*r)+(i+i*r)*r

For any given day d, and existed cases x and infection rate r

Day	existed cases	New cases	Total cases
d	x	x*r	x+x*r=x*(r+1)
d+1	x*(r+1)	x*(r+1)*r	x*(r+1)+x*(r+1)*r

We can observe that the number of total cases of any day d isonly related to that of day $d-1$. And infection rate r is constant.

Day	Total cases
d	X
d+1	X*(r+1)

So the update of total cases is

total cases= total case*(r+1)

Then the structure of the loop is clear: in each round of the loop, we simulate the infection for a new day. And loop condition is that the spread will continue until the entire population is infected

```
while (total cases<population){
        total cases= total case*(r+1)
}
```

However, the problem requires to print the day on which the *entire* population will be infected. So, it is equivalent to track how many loops it takes to infect the entire population. A convenient way is initializing an integer variable and increment it each time before infection parts are executed.

```
int day=1;

while (total cases<population){

        day++;

        total cases= total case*(r+1);

}
```

Or increment it each time after infection parts are executed

```
int day=1;

while (total cases<population){

        total cases= total case*(r+1);

        day++;

}
```

Check implementations for solutions (both “while” and “for” loops are implemented). “do…while” is not recommended in this case because we have to deal with special cases and write some redundant lines of code (we cannot simulate what happens in day one). “while” and “for” loops offer a powerful logic to skip “do” part if the pre-requisite is not true.

Optimization:

For some loop-based mathematical problems, we can work out an equation that calculate the answer directly. And we will see how to find it.

Given initial infection number $i \in N^+$, infection rate $r \in N^+$ at day 1 and assume population is infinity, we can calculate total cases $T(\mathrm{d})$ till any day $d \geq 1$ with only one equation

$$T(d) = i * (r + 1)^{d\text{-}1}$$

Thus, if population p is finite, then we can find out the minimum number of d s.t. $T(d) \geq p$. Solve it and we have

$$d \geq \log_{r+1}\frac{p}{l} + 1$$

However, the first term of right part is not always an integer. Thus we take its ceil because we always need one more day to infect left healthy people even if our infection capacity exceeds left heathy people. Thus

$$d = \left\lceil \log_{r+1}\frac{p}{l} \right\rceil + 1$$

Note that java does not provide logarithmic calculation with arbitrary bases. But we can use a property of logarithmic

$$\log_a b = \frac{\log b}{\log a}$$