class Person() {

int arrivalTime;

int timeUse;

//constructor

//getters

}

class arrivalComparator implements Comparator <Person> {

public int compare(Person p1, Person p2) { //sort by earliest is the first element

(p1.getArrivalTime() > p2.getArrivalTime()) {

return 1;

}

}

}

public void Main () {

int count = 0;

int saverTime = read from input;

PriorityQueue<Integer> inUse;

PriorityQueue<Integer> unUsed;

For each worker:

Arraylist<Person>list.add(worker);

list.sort(arrivalComparator);

for item in list {

Person p = list.get(i);

while ((!inUse.isEmpty()) && arrivalTime>inUse.peek()) { //ws curr ununsed

unUsed.offer(inUse.poll()+saverTime);

}

while((!unUsed.isEmpty()) && arrivalTime > unUsed.peek())) { //ws alr locked back

unUsed.poll(); //remove them

}

if (!unUsed.isEmpty()) { //after removing all the locked ones, still got any that is available? Will take the one that is gonna lock back the soonest (i.e. the top element of the heap)

unUsed.poll();

count++; //saved penelop from unlocking this one.

}

inUse.offer(p.getArrivalTime()+p.getTimeUse()); //finally, assign new worker to a ws

}

println(count);

)