[**Exclusive Time of Functions**](https://leetcode.com/problems/exclusive-time-of-functions/)

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.Stack;

**public** **class** ExclusiveTimeFunctions {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

List<String> logs = **new** ArrayList<>();

logs.add("0:start:0");

logs.add("0:start:2");

logs.add("0:end:5");

logs.add("0:start:6");

logs.add("0:end:6");

logs.add("0:end:7");

**int**[] result = *exclusiveTime*(1 , logs);

**for**(**int** i : result) {

System.***out***.print(i + " ");

}

}

**public** **static** **int**[] exclusiveTime(**int** n, List<String> logs) {

**if**(logs == **null** || logs.size() == 0) {

**return** **new** **int**[]{};

}

**int**[] result = **new** **int**[n];

Stack<Integer> stack = **new** Stack<>();

String[] log = logs.get(0).split(":");

stack.push(Integer.*parseInt*(log[0]));

**int** prev = Integer.*parseInt*(log[2]);

**int** i = 1;

**while**(i < logs.size()) {

log = logs.get(i).split(":");

**if**(log[1].contains("start")) {

**if**(!stack.isEmpty()) {

result[stack.peek()] += Integer.*parseInt*(log[2]) - prev;

}

stack.push(Integer.*parseInt*(log[0]));

prev = Integer.*parseInt*(log[2]);

}

**else** {

result[stack.peek()] += Integer.*parseInt*(log[2]) - prev + 1;

stack.pop();

prev = Integer.*parseInt*(log[2]) + 1;

}

i++;

}

**return** result;

}

}

Time complexity : O(m) , m is size of the given log list

Space Complexity : O(n) ,n is no of functions