The accounting department turned to the IT department with a request to help with the financial report on the work of teams on the vendor side. It is known that the teams consist of a different number of specialists in different categories. It is also known that each category of specialists has its own fixed salary after taxes and this tax rate. Accounting must calculate the cost of the services for each specialty and the entire team. Your team leader has prepared a function template that will perform the task. You need to implement this function. Please note that some important hints are described in the comments to the code examples. See details below

```
function calculateTeamFinanceReport(salaries, team)
```

where `salaries` is an object with more information about salaries and taxes by specialist categories; minimum specializations amount is 1, maximum up to 10; see template below

```
{
    Progger: { // specialization type 'Progger'
        salary: 1000, // salary after tax; should be integer; min: 100, max: 100000
        tax: "15%" // tax percent; presented as string with template `{tax}%` where 'tax' is integer;  min: "0%", max: "99%"
    },
    Tester: {
        salary: 1000,
        tax: "10%"
    }
}
```

where `team` is an array of objects represented by member of the team with name and specialization; minimal team size is 1, maximum up to 100; see template below

```
[
    {
        name: "Masha", // name of team member
        specialization: "Progger" // specialization should be picked from `salaries` otherwise member should be ignored in report
    },
    {
        name: "Vasya",
        specialization: "Tester"
    },
    {
        name: "Taras",
        specialization: "Tester"
    },
]
```

the function should return the report object following the next template

```
{
    totalBudgetTeam: 3398, // total salaries with tax of entire team; should be integer (truncate the fractional part after all calculations)
    totalBudgetProgger: 1176, // total salaries with tax for all members by 'Progger' specialization; should be integer (truncate the fractional part after all calculations)
    totalBudgetTester: 2222, // total salaries with tax for all members by 'Tester' specialization; should be integer (truncate the fractional part after all calculations)
}
```

See full example #1 below

```
const salaries1 = {
   Manager: { salary: 1000, tax: "10%" },
   Designer: { salary: 600, tax: "30%" },
   Artist: { salary: 1500, tax: "15%" },}
const team1 = [
   { name: "Misha", specialization: "Manager" },
   { name: "Max", specialization: "Designer" },
   { name: "Vova", specialization: "Designer"},
   { name: "Leo", specialization: "Artist"},]
const financeReport1 = calculateTeamFinanceReport(salaries1, team1)
console.log(JSON.stringify(financeReport1))
/* see in console
{
   "totalBudgetTeam":4590, // total budget does not match the sum of specializations due
to truncation of the fractional part
   "totalBudgetManager":1111,
   "totalBudgetDesigner":1714,
   "totalBudgetArtist":1764,
}
*/
```

See full example #2 below

```
const salaries2 = {
   TeamLead: { salary: 1000, tax: "99%" },
   Architect: { salary: 9000, tax: "34%" },}
const team2 = [
   { name: "Alexander", specialization: "TeamLead" },
   { name: "Gaudi", specialization: "Architect" },
   { name: "Koolhas", specialization: "Architect" },
   { name: "Foster", specialization: "Architect" },
   { name: "Napoleon", specialization: "General" },]
const financeReport2 = calculateTeamFinanceReport(salaries2, team2)
console.log(JSON.stringify(financeReport2))
/* see in console
{"totalBudgetTeam":140909,"totalBudgetTeamLead":100000,"totalBudgetArchitect":40909}
*/
```