You can view this report online at : https://www.hackerrank.com/x/tests/584397/candidates/13651447/report

Full Name:	Alexandr Romanov
Email:	rmnff.dev@yandex.ru
Test Name:	**Backend challenge from 6nomads**
Taken On:	20 Mar 2020 19:36:29 +03
Time Taken:	82 min 27 sec/ 120 min
Work Experience:	5 years
Invited by:	Denis
Invited on:	20 Mar 2020 19:26:38 +03
Tags Score:	Implementation 75/75 Medium 75/75 Strings 75/75

100%

75/75

scored in **Backend challenge from 6nomads** in 82 min 27 sec on 20 Mar 2020 19:36:29 +03

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Sort the Events > **Coding**	1 hour 15 min 40 sec	75/ 75	✓

QUESTION 1

Correct Answer

Score 75

Sort the Events > Coding

Implementation | Strings | Medium

QUESTION DESCRIPTION

Two software development teams had a meeting and a record of events was created. There are 4 possible events, gather requirements (G), merge requirements (Y), redesign (R), and solve (S).

- The first three events (G, Y, R) are represented as `employee-name time event-name`
- The last event (S) is represented as `employee-name time event-name second-employee-name`.
- The time is represented in minutes from the start of the meeting.
- The meeting is divided into two halves of 45 minutes each.
- There can be two representations of time:
  - regular time is represented as a single integer, `time`, for example `45`
  - additional time is represented as `time+extra-time` for example `45+2`

Note: The extra time is always considered to have occurred before the events of the next half, so, `45+2` happened earlier than `46`.

Merge the events for each development team into a single meeting event with the team's name in front, sorted by time and event in the order G, Y, R, S. In the case of the same event happening at the same time, output should be sorted alphabetically based on the team's name and then the employees's name.

Example

The first team name, *team1 = "edc"* with events recorded as:

events1 = ['alex 12 G',
'sam 43 Y']

The second team name, *team2 = "cde"* with events recorded as:

events2 = ['kris 45+1 S avery',
'robin 46 G']

The chronological order of events is:

edc alex 12 G
edc sam 43 Y
cde kris 45+1 S avery
cde robin 46 G

Function Description

Complete the function *getEventsOrder* in the editor below.

getEventsOrder has the following parameter(s):

string team1: a string, the name of the first team
string team2: a string, the name of the second team
string events1[n1]: an array of strings that describe the events of the first team
string events2[n2]: an array of strings that describe the events of the second team

Returns

string[n]: an array of strings

Constraints

- *0 ≤ length of events1, events2 ≤ 20*
- *0 < length of team's name ≤ 50*
- *0 < length of player's name ≤ 50*
- The team names and employee names will consist only of lowercase English letters and blank space
- time will be between 0 to 90
- extra time will always be written after `+`

▼ Input Format For Custom Testing

The first line contains a string, the name of *team1*.

The second line contains a string, the name of *team2*.

The next line contains an integer, *n1*, the number of elements in *events1*.

Each line *i* of the *n1* subsequent lines (where *0 ≤ i < n1*) contains a string that describes *events1[i]*.

The next line contains an integer, *n2*, denoting the number of elements in *events2*.

Each line *j* of the *n2* subsequent lines (where *0 ≤ j < n2*) contains a string that describes *events2[j]*.

▼ Sample Case 0

Sample Input For Custom Testing

```
STDIN            Function
-----            -----
abc          →   team1 = 'abc'
cba          →   team2 = 'cba'
```

```
2             →  events1[] size n1 = 2
mo sa 45+2 Y  →  events1 = ['mo sa 45+2 Y', 'a 13 G']
a 13 G
2             →  events2[] size n2 = 2
d 23 S f      →  events2 = ['d 23 S f', 'z 46 G']
z 46 G
```

Sample Output

```
abc a 13 G
cba d 23 S f
abc mo sa 45+2 Y
cba z 46 G
```

Explanation

The first event happens at the 13^{th} minute (G), then the next event happens at the 23^{rd} minute (S). The next one occurs at the $45+2^{nd}$ minute (Y), and the last event at the 46^{th} minute (G).

▼ Sample Case 1

Sample Input For Custom Testing

```
STDIN                        Function
-----                        --------
nolh                    →    team1 = 'nolh'
uzrdrrc slcpx           →    team2 = 'uzrddrc slcpx'
3                       →    events1[] size n1 = 3
inmuucz jzbkica 70 Y    →    events1 = ['inmuucz jzbkica 70 Y', 'ton wfnt
10 S inmuucz jzbkica', 'ecya kgvgy 20 S fkfk fuiyb senmofw']
ton wfnt 10 S inmuucz jzbkica
ecya kqvqy 20 S fkfk fuiyb senmofw
3                       →  events2[] size n2 = 3
mysior pqfcz bxlnpn 49 G  →  events2 = ['mysior pqfcz bxlnpn 49 G',
'mysior pqfcz bxlnpn 18 G', 'enc otagavd oevfg 68 Y']
mysior pqfcz bxlnpn 18 G
enc otagavd oevfg 68 Y
```

Sample Output

```
nolh ton wfnt 10 S inmuucz jzbkica
uzrdrrc slcpx mysior pqfcz bxlnpn 18 G
nolh ecya kqvqy 20 S fkfk fuiyb senmofw
uzrdrrc slcpx mysior pqfcz bxlnpn 49 G
uzrdrrc slcpx enc otagavd oevfg 68 Y
nolh inmuucz jzbkica 70 Y
```

Explanation

The times for each event are all different, so these can be sorted based only on time.

CANDIDATE ANSWER

Language used: **JavaScript (Node.js)**

```
/*
 * Complete the 'getEventsOrder' function below.
 *
 * The function is expected to return a STRING_ARRAY.
 * The function accepts following parameters:
 *  1. STRING team1
 *  2. STRING team2
 *  3. STRING_ARRAY events1
 *  4. STRING_ARRAY events2
 */

function getEventsOrder(team1, team2, events1, events2) {
    const table = {
```

```
        "G": 1,
        "Y": 2,
        "R": 3,
        "S": 4
    };
    // Write your code here
    const events1numbers = [];
    events1.map((event, i) => {
        const split = event.split(' ');
        split.map((s, y) => {
            const probnum = s.split('+');
            if(probnum.length != 1) {
                return events1numbers.push({ num:
Number.parseInt(probnum[0]), pos: i, team: 1, additional:
Number.parseInt(probnum[1]), operation: table[split[y + 1]] })
            }
            if(!Number.isNaN(Number.parseInt(s))) {
                return events1numbers.push({ num: Number.parseInt(s), pos: i,
team: 1, additional: null, operation: table[split[y + 1]] });
            }
        })
    });
    const events2numbers = [];
    events2.map((event, i) => {
        const split = event.split(' ');
        split.map((s, y) => {
            const probnum = s.split('+');
            if(probnum.length != 1) {
                return events2numbers.push({ num:
Number.parseInt(probnum[0]), pos: i, team: 2, additional:
Number.parseInt(probnum[1]), operation: table[split[y + 1]] })
            }
            if(!Number.isNaN(Number.parseInt(s))) {
                return events2numbers.push({ num: Number.parseInt(s), pos: i,
team: 2, additional: null, operation: table[split[y + 1]] });
            }
        })
    });
    const eventsAllNumbers = events1numbers.concat(events2numbers);
    eventsAllNumbers.sort((a,b) => {
        if(a.num == b.num && a.additional != null && b.additional != null) {
            if((a.num + a.additional) == (b.num + b.additional)) {
                return a.operation - b.operation;
            }
            return (a.num + a.additional) - (b.num + b.additional);
        } else if(a.num == b.num && a.additional != null) {
            if((a.num + (a.additional || 0)) == (b.num + b.additional || 0))
{
                return a.operation - b.operation;
            }
            return (a.num + (a.additional || 0)) - (b.num + b.additional ||
0);
        } else if(a.num == b.num && b.additional != null) {
            if((a.num + (a.additional || 0)) == (b.num + b.additional)) {
                return a.operation - b.operation;
            }
            return (a.num + (a.additional || 0)) - (b.num + b.additional);
        } else if(a.num == b.num) {
            if(a.operation == b.operation) {
                let specteam1 = null;
                let specevent1 = null;
                let specteam2 = null;
                let specevent2 = null;
                if(a.team == 1) { specteam1 = team1; specevent1 =
```

```
78 events1[a.pos]; } else { specteam1 = team2; specevent1 = events2[a.pos]; };
79                 if(b.team == 1) { specteam2 = team1; specevent2 =
80 events1[b.pos]; } else { specteam2 = team2; specevent2 = events2[b.pos] };
81                 console.log(`${specteam1} ${specevent1}`, `${specteam2}
82 ${specevent2}`);
83                 if(`${specteam1} ${specevent1}` > `${specteam2}
84 ${specevent2}`) return 1;
85                 if(`${specteam1} ${specevent1}` < `${specteam2}
86 ${specevent2}`) return -1;
                   return 0;
               }
               return a.operation - b.operation;
           } else return a.num - b.num;
       });
       const sortedEvents = eventsAllNumbers.map(event => {
           if(event.team == 1) return `${team1} ${events1[event.pos]}`;
           else return `${team2} ${events2[event.pos]}`;
       });
       return sortedEvents;
}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Test Case 0	Easy	Sample case	Success	1	0.158 sec	30.4 KB
Test Case 1	Easy	Sample case	Success	1	0.1653 sec	30.5 KB
Test Case 2	Easy	Sample case	Success	1	0.1542 sec	30.4 KB
Test Case 3	Easy	Sample case	Success	8	0.1724 sec	30.4 KB
Test Case 4	Easy	Sample case	Success	8	0.2214 sec	30.4 KB
Test Case 5	Easy	Hidden case	Success	8	0.1687 sec	30.4 KB
Test Case 6	Easy	Hidden case	Success	8	0.1563 sec	30.5 KB
Test Case 7	Easy	Hidden case	Success	8	0.1615 sec	30.4 KB
Test Case 8	Easy	Hidden case	Success	8	0.1976 sec	30.4 KB
Test Case 9	Easy	Hidden case	Success	8	0.1882 sec	30.4 KB
Test Case 10	Easy	Hidden case	Success	8	0.2648 sec	30.4 KB
Test Case 11	Easy	Hidden case	Success	8	0.1691 sec	30.3 KB

No Comments