# Programming Fundamentals Mid Exam Retake 07 April 2020

## Problem 3. Moving Target

You are at the shooting gallery again and you need a program that helps you keep track of moving targets. On the first line, you will receive a **sequence of targets with their integer values**, split by a **single space**. Then, you will start receiving **commands for manipulating the targets**, until the **"End"** command. The commands are the following:

* **Shoot {index} {power}**
  + Shoot the target at the index, **if it exists** by **reducing** its **value** by the **given** **power** (**integer value**).A target is considered **shot** when **its value reaches 0**.
  + Remove the target, **if it is shot**.
* **Add {index} {value}**
  + Insert a target with the received value at the received **index, if it exist**. If not, print: **"Invalid placement!"**
* **Strike {index} {radius}**
  + Remove the **target at the given index** and the **ones before and after it depending on the radius, if such exist. If any of the indices in the range is invalid print:**

**"Strike missed!" and skip this command.**

**Example:** **Strike 2 2**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | {radius} | {radius} | {strikeIndex} | {radius} | {radius} |  |  |

* **End**
  + Print the sequence with targets in the following format:

**{target1}|{target2}…|{targetn}**

### Input / Constraints

* On the **first line** you will receive **the sequence of targets** – **integer values [1-10000]**.
* On the **next lines,** until the **"End"** will be receiving the command described above – **strings**.
* There will never be a case when **"Strike"** command would empty the whole sequence.

### Output

* Print the appropriate message in case of **"Strike"** command if necessary.
* In the end, print the sequence of targets in the format described above.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 52 74 23 44 96 110  Shoot 5 10  Shoot 1 80  Strike 2 1  Add 22 3  End | Invalid placement!  52|100 | The first command is "**Shoot**", so we reduce the target on **index** **5**, which is valid, with the given **power** – **10**.  Then we receive the same command but we need to reduce the target on the 1st index, with power 80. The value of this target is 74, so it is considered shot and we **remove** it.  Then we receive the "**Strike**" command on the 2nd index and we need to check if the range with radius 1 is valid:  **52 23 44 96 100**  And it is, so we **remove** the targets.  At last we receive the "**Add**" command, but the index is **invalid** so we print the appropriate **message** and in the end we have the following result:  **52|100** |
| 1 2 3 4 5  Strike 0 1  End | Strike missed!  1|2|3|4|5 |  |

function movingTarget(arr) {

    arr.pop()

    let targets = arr.shift().split(' ').map(Number)

    while (arr.length > 0) {

      const [command, index, power] = arr.shift().split(' ').map(el => isNaN(el) ? el : Number(el))

      if (command === 'Shoot') {

        if (index >= 0 && index < targets.length)

          targets[index] = targets[index] - power

        if (targets[index] <= 0)

          targets.splice(index, 1)

      }

      if (command == 'Strike') {

        let radius = power

        if (index + radius < targets.length && index - radius >= 0)

          targets.splice(index - radius, radius \* 2 + 1)

        else

          console.log('Strike missed!')

      }

      if (command === 'Add') {

        if (index >= 0 && index < targets.length)

          targets.splice(index, 0, power)

        else

          console.log('Invalid placement!')

      }

    }

    console.log(targets.join('|'))

  }

movingTarget(['52 74 23 44 96 110', 'Shoot 5 10', 'Shoot 1 80', 'Strike 2 1', 'Add 22 3', 'End'])