Problem 3 – Listy

Bai Ivan also known as @bivan27 is a famous blogger and developer. Because he is Bulgarian, he is not a big fan of buying software, so he doesn't have a legal version of Microsoft Office. Also because of his neighbour who doesn't like him and sends him bad boys from GDBOP(ГДБОП) every week to check his PC for illegal software, @bivan27 can't use Excel.

One day @bivan27 decided to make his own programming language called **Listy** which will do Excel's calculations with a list of numbers. His conception for now is very simple. He can assign lists to a variables, get the **min** or **max** value from a list, get **average** or **sum** the elements of the list. Each of the functions gets as parameters list of numbers or variables in square brackets."**def**" is a keyword used to define a variable. Here are some syntax examples.

def var1 [1, 2, 6, 8] //assign list to the variable **var1**
def var2 sum[1, 5, -10, 20] //assign result of the operation sum to a variable var2. Result is 16
def var3 max[5, 2, 4, **var2**, 2] //assign the max number of the list to **var3.** Result is 16 (comes from **var2**)
	`def var4 min[var1, 6, 50]`
	`//var4=1 (comes from` **`var1`**`)`
	`def var5 avg[var1]`
	`//var5=`**`4`**
	`(1+2+6+8=17/4=`**`4.25`**`)`
	`(The avg returns number without the remainder!)`
		`def var6 sum[var1, var1, 1]`
		`//var6=35 (17+17+1)`

Everything looks great? Right? But @bivan27 has some problem with his dog "Sharo" and he doesn't have the time to make an interpreter for Listy. Help him by **writing Listy interpreter in JavaScript**, because tomorrow morning he has meeting with new investors, who wants to use his project for calculation in Boza's production.

NOTE: There could be more than one or no whitespace between the characters. For example
def   varName   sum [2,3,12 ,   4, 1   ) //Also has to return 22

Also you can use old functions in the definitions of the new one. The interpreter should run code in this format:

```
def var1[1, 2, 3 ,4]
def var2 sum[var1, 20, 70] //var2=100
def var3 min[var1] // var3=1
avg[var2, var3] //the result is 50
```

NOTE: There will be only a sequence of numbers and variables in the definition of a new variable.
NOTE: There will be no nested commands in the given command
Example: Command can be "def var sum[1,2,3]" but **it won't be** "def var sum[1,2,3, min[var0, 3,-5,2]]"

You are given an array of strings (commands). Execute all the commands and **print the result only from the last line!**

- If you meet a variable in a command it'll be always defined in some of the lines before!
- "- 5" is not valid number but "-5" is.
- Variable's names are case sensitive.
- Variables cannot be overwritten.
- Each string will be a valid Listy command.

- Variable can contains definition of a number or list of numbers
- If there is no operation on the last line, command will looks like “[var1]”. Otherwise if there is a final command it’ll be in format: “sum[var1,var2]” (or other operation)

Write method **Solve** that accepts the commands as array and prints the result of the last command.

Input

The method **Solve** accepts an array of strings. (Example: arr=[“command1”,”command2”,”command3”])

Output

Your method should return a single line - the result of the last command

Example code

```
function Solve(params) {
	var answer = 0;
	// Your code here...
	return answer;
}
```

Constraints

- Array size will be between 1 and 500 elements.
- Each element of the Array will be string containing valid command.
- Each list will be between 1 and 20 variables/numbers.
- Allowed working time for your program: 0.2 seconds. Allowed memory: 16 MB.

Examples (each line represents an element (string) from the only argument of Solve)

Example input	**Example output**

def func sum[5, 3, 7, 2, 6, 3]
def func2 [5, 3, 7, 2, 6, 3]
def func3 min[func2]
def func4 max[5, 3, 7, 2, 6, 3]
def func5 avg[5, 3, 7, 2, 6, 3]
def func6 sum[func2, func3, func4]
sum[func6, func4]
	42

Example input	**Example output**

def func sum[1, 2, 3, -6]
def newList [func, 10, 1]
def newFunc sum[func, 100, newList]
[newFunc]
	111