problem	zidarska	malloc	jaja

source file	`zidarska.pas`
	`zidarska.c`
	`zidarska.cpp`
		`malloc.pas`
		`malloc.c`
		`malloc.cpp`
			`jaja.pas`
			`jaja.c`
			`jaja.cpp`

input data	stdin		library
output data	stdout		library
memory limit (heap)	16 MB		
memory limit (stack)	2 MB		
time limit (pentium4 1.6 ghz)	1 second		
points	**50**	**70**	**80**
	200		

Mirko wants to download his favorite song from the internet.

The song is divided into segments that must be downloaded in fixed order. We know the **length** and **download time** of each segment (both are expressed in seconds).

If it's possible, he wants to start listening to his song **as soon as possible**, even before it's downloaded, with **no interruptions** (no waiting for some segment to be downloaded in the middle of the song). He can start listening to a segment only after it has been **completely downloaded**.

Write a program that will determine the **minimal** number of seconds (from the moment that download starts) after which Mirko can start listening.

input data

In the first line there is an integer N, $1 \leq N \leq 100{,}000$, number of segments.

In each of the following N lines, there are two integers, D and V, $1 \leq D,V \leq 1000$. These numbers represent segments in the order they have to be downloaded. D is length, V is download time.

output data

In the first and only line write the number of seconds from the text.

examples

input

```
4
2 1
1 5
3 3
2 4
```

output

```
7
```

input

```
5
1 1
1 2
3 1
2 1
3 3
```

output

```
2
```

input

```
7
2 1
2 4
1 2
2 1
3 2
3 1
1 3
```

output

```
3
```

Write a program that will simulate the execution of memory allocation commands.

Memory is a sequence of **100,000 continuous memory locations**, numbered from 1 to 100,000.

In the beginning, all locations are free.

Commands that can occur are:

1.) `var=malloc(size);`

This function finds the first `size` (100 ≤ `size` ≤ 100,000) **continuous free locations**, and allocates them. Return value of this function is either **the address of the first allocated location** or **0** if there is **no such sequence**.

2.) `free(var);`

This function frees all the memory locations assigned to the variable `var` (by previous use of `malloc`) and sets the value of `var` to **0**.

If the value of variable `var` is already **0**, nothing happens.

3.) `print(var);`

This function prints the value of variable `var`.

Every command line ends with a **semicolon** ('**;**').

Variables are strings of **exactly four** small letters of the english alphabet ('a'...'z').

Number of different variables will be **less than or equal to 1000**.

All variables are set to **0** in the beginning.

input data

In the first line there is an integer N, $1 \leq N \leq 100{,}000$, number of commands (**at least** one of the commands will be '`print`').

In each of the following N lines there is one command, in order of their execution.

output data

Write the results of all '`print`' commands, in order of their execution, each one in separate line.

examples

input

```
3
mama=malloc(140);
tata=malloc(120);
print(tata);
```

output

```
141
```

input

```
5
aabb=malloc(50001);
bbaa=malloc(50000);
print(aabb);
free(aabb);
print(bbaa);
```

output

```
1
0
```

input

```
5
baka=malloc(214);
baka=malloc(123);
free(baka);
deda=malloc(100);
print(deda);
```

output

```
215
```

Mirko got a job in an agricultural company, where he runs tests concerning the quality of fresh eggs. Every day, he gets **12 identical eggs** and tests their resistance to falls from great heights.

Resistance is a **positive integer less than or equal to 1,000,000**. If we drop the same egg several times, it will break if the **sum of all heights** (expressed in millimeters) it was dropped from, is **greater** than its resistance.

For example, if we take an egg with resistance 6 and drop it from heights 5 and then 2, it will break on the 2nd fall. If we take a new egg with resistance 10 and drop it from heights 5, 5, and 10, it will break on the 3rd fall.

In the beginning, Mirko has 12 brand new eggs with **the same resistance**. In each step he can drop an unbroken egg from some height. He must determine the resistance using **no more than 100 drops**.

Write a program that will help him.

library

For solving this task, you are given a library file `jajalib` which contains three functions:

`Init` - this function you must call **exactly once**, **at the beginning** of your program, without parameters. This function returns no value.

```
procedure Init;
void Init(void);
```

`Baci` - this function is called with two parameters, first one is the number of the egg you want to throw ($1 \leq$ `egg` ≤ 12), and the second is the height ($1 \leq$ `height` $\leq$ 1,000,000) from which you want to drop it from.

Function returns **1** if the egg **breaks** and **0** if it **doesn't**. Not respecting the given limitations on the parameters or dropping of already broken egg will result in the loss of points.

```
function Baci(egg, height : longint) : longint;
int Baci(int egg, int height);
```

`Rjesenje` - this function you need to call at the end of your program. The only parameter is the resistance of eggs. This function returns no value, and **ends the execution** of your program **regularly**.

```
procedure Rjesenje(resistance : longint);
void Rjesenje(int resistance);
```