zadatak	zidarska	malloc	jaja

izvorni kôd	`zidarska.pas`
	`zidarska.c`
	`zidarska.cpp`
		`malloc.pas`
		`malloc.c`
		`malloc.cpp`
			`jaja.pas`
			`jaja.c`
			`jaja.cpp`

ulazni podaci	standardni ulaz		biblioteka
izlazni podaci	standardni izlaz		biblioteka
memorijsko ograničenje (heap)	16 MB		
memorijsko ograničenje (stack)	2 MB		
vremensko ograničenje (pentium4 na 1.6 ghz)	1 sekunda		
broj bodova	**50**	**70**	**80**
	200		

Narodna pjesma "**Zidarska**" je već dulje vrijeme jako popularna.

Mirko još nije čuo tu pjesmu pa ju je odlučio skinuti s interneta. Kako je ona stvarno megapopularna, brzina skidanja nije baš prevelika, a Mirko bi je htio **što prije** čuti.

Pjesma je podijeljena na dijelove koji se redom skidaju, a **duljina** svakog dijela i **brzina** skidanja tog dijela pjesme je poznata.

Kako Mirko umire od želje da je **što prije** čuje, on pjesmu želi početi slušati i prije nego što je u potpunosti skinuta, ali pod uvjetom da mu pjesma **od početka do kraja** svira **bez prekida**. Neki dio pjesme Mirko može početi slušati tek kada se taj dio **u potpunosti** skine.

Napišite program koji će izračunati nakon koliko vremena od početka skidanja pjesme Mirko može početi slušati "Zidarsku".

ulazni podaci

U prvom retku se nalazi cijeli broj N, $1 \leq N \leq 100{,}000$, broj dijelova pjesme.

U svakom od sljedećih N redaka se nalaze po dva cijela broja D i V, $1 \leq D,V \leq 1000$. Ti brojevi predstavljaju opise dijelova pjesme, redom od početka prema kraju pjesme.

Broj D označava duljinu tog dijela pjesme (u sekundama), a broj V vrijeme potrebno da se taj dio skine s interneta (u sekundama).

izlazni podaci

U prvi i jedini redak treba ispisati traženo vrijeme iz teksta zadatka (u sekundama).

test primjeri

ulaz

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

izlaz

```
7
```

ulaz

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

izlaz

```
2
```

ulaz

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

izlaz

```
3
```

Napišite program koji će simulirati izvršavanje naredbi za rezerviranje i oslobađanje memorije.

Memorija računala je niz od **100,000 uzastopnih memorijskih lokacija**, redom označenih brojevima od 1 do 100,000.

Na početku su sve lokacije **slobodne**.

Naredbe koje se mogu pojaviti su:

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

Ova naredba pronalazi prvi niz od `velicina` ($100 \leq$ `velicina` $\leq 100{,}000$) **uzastopnih slobodnih** memorijskih lokacija i rezervira ih. Funkcija `malloc` vraća adresu **prve rezervirane lokacije** ili **0** ako **ne postoji** niz od `velicina` uzastopnih slobodnih memorijskih lokacija.

2.) `free(var);`

Ova naredba oslobađa memorijske lokacije dodijeljene varijabli `var` (prethodnim rezerviranjem funkcijom `malloc`) i postavlja vrijednost varijable `var` na **0**.

Ako je vrijednost varijable `var` jednaka **0**, ništa se ne događa.

3.) `print(var);`

Ova naredba ispisuje vrijednost varijable `var`.

Svaka naredba završava znakom **';'** (**točkazarez**).

Varijable su nizovi sastavljeni od **točno 4 znaka**, a jedini dozvoljeni znakovi su mala slova engleske abecede ('a'...'z').

Ukupni broj različitih varijabli će biti **manji ili jednak od 1000**.

Sve varijable su na početku inicijalizirane na vrijednost **0**.

ulazni podaci

U prvom retku se nalazi cijeli broj N, $1 \leq N \leq 100{,}000$, broj naredbi (**barem jedna** naredba će biti 'print').

U svakom od sljedećih N redaka se nalazi jedna naredba, redom izvršavanja.

izlazni podaci

Potrebno je ispisati rezultate svih 'print' naredbi, svaki rezultat u poseban red.

test primjeri

ulaz

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

izlaz

```
141
```

ulaz

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

izlaz

```
1
0
```

ulaz

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

izlaz

```
215
```

jaja

Mirko je odustao od studiranja na FER-u te se preko veze zaposlio u lokalnom poljoprivrednom kombinatu gdje radi na testiranju kvalitete svježih jaja. Točnije, Mirko svakog dana dobije **12 komada** novih identičnih jaja i njegov zadatak je da ispita kakva je otpornost tih jaja prilikom pada s visine na tvrdu površinu.

Otpornost jaja je cijeli broj **veći ili jednak od 1** i **manji ili jednak od 1,000,000**. Ako neko jaje bacimo nekoliko puta na pod s raznih visina ono će puknuti u onom trenutku kada **zbroj svih visina** (u milimetrima) s kojih je to jaje bačeno bude **veći** od otpornosti.

Npr. ako uzmemo novo jaje otpornosti 6 i ako ga bacimo s 5 i zatim sa 2 milimetra, ono će puknuti prilikom drugog bacanja. Ako pak uzmemo novo jaje otpornosti 10 pa ga bacimo s 5, pa opet s 5, i na kraju s 10 milimetara, ono će puknuti tek prilikom trećeg bacanja.

Na početku Mirko ima na raspolaganju 12 novih jaja i sva jaja imaju **istu** otpornost O. U svakom koraku Mirko može baciti jedno nerazbijeno jaje s neke visine. Potrebno je odrediti otpornost O pomoću **najviše 100 bacanja**.

Napišite program koji će pomoći Mirku testirati jaja.

biblioteka

Za rješavanje zadatka na raspolaganju je biblioteka funkcija `jajalib` koja sadrži tri funkcije:

`Init` - ovu funkciju je potrebno pozvati **točno jednom** i to **na početku** vašeg programa, bez argumenata. Ova funkcija ne vraća nikakvu vrijednost.

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

`Baci` - ova funkcija se poziva s dva argumenta, prvi argument je redni broj jajeta ($1 \leq$ `jaje` ≤ 12), a drugi argument visina ($1 \leq$ `visina` $\leq$ 1,000,000) s koje želimo baciti to jaje. Funkcija vraća **1** ako se jaje pri tom bacanju **razbije**, a **0** ako se jaje **ne razbije**. Nepoštivanje zadanih ograničenja na argumente ili bacanje već prije razbijenog jajeta rezultirat će gubitkom bodova.

```
function Baci(jaje, visina : longint) : longint;
int Baci(int jaje, int visina);
```

`Rjesenje` - ovu funkciju je potrebno pozvati **na kraju** vašeg programa; kao argument prosljeđujete traženu otpornost jaja. Ova funkcija ne vraća nikakvu vrijednost i **regularno završava izvođenje** vašeg programa.

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

Upute za natjecatelje u Pascal-u

Na početku vašeg programa mora se nalaziti naredba `'uses jajalib'`.

Također, potrebno je preuzeti datoteke `jajalib.o` i `jajalib.ppu` s evaluacijskog sustava.

Upute za natjecatelje u C/C++

Potrebno je preuzeti datoteke `jajalib.h` i `jajalib.o` s evaluacijskog sustava.

U RHIDE-u otvorite projekt (Project->Open) i nazovite ga `jaja`. U njega naredbom

Project->AddItem dodajte datoteke `jaja.c` odnosno `jaja.cpp` (vaš program) i `jajalib.o`.

Na početku vašeg programa mora se nalaziti naredba `'#include "jajalib.h"'`.

Važno: nemojte koristiti naredbu Compile->BuildAll (jer ona briše datoteku `jajalib.o`).

Testiranje

Vaš program možete testirati tako da na standardni ulaz unesete vrijednost otpornosti jaja, a biblioteka će na standardni izlaz ispisivati opširne poruke o rezultatima bacanja jajeta.