Feira de Bactérias

Nome do arquivo fonte: `bacterias.c`, `bacterias.cpp`, *ou* `bacterias.pas`

Bruno é um biólogo apaixonado por sua profissão. Sua especialidade é estudar o comportamento de bactérias. Por isso, ele possui em seu laboratório centenas de colônias de diferentes tipos desses microorganismos.

Nesta semana ele viu o anúncio de um evento inusitado: uma feira de bactérias. Nessa feira, vários fornecedores estarão vendendo diferentes tipos de bactérias. Cada tipo de bactéria é vendido em uma placa de vidro, já preparada para a formação de uma colônia de bactérias. Cada placa de vidro é vendida com apenas uma bactéria inicialmente.

Bruno deu uma olhada no catálogo com os tipos de bactérias que estarão à venda na feira, e notou algumas coisas interessantes:

- Todos os tipos de bactérias à venda terão o mesmo preço.
- Todas as bactérias (de todos os tipos) se subdividem todas as noites para gerar outras bactérias. Por exemplo, a bactéria da colônia de tipo X se subdivide em 2 outras bactérias todas as noites. Assim, no primeiro dia teremos só uma bactéria na colônia. No dia seguinte, teremos 2, e no próximo, 4. A quantidade de divisões de uma bactéria depende do seu tipo.
- O crescimento da colônia cessa após um determinado número de dias, por causa da escassez de alimento. A quantidade de dias em que uma colônia cresce depende do tipo de bactéria.

É final de mês e Bruno já gastou quase todo o seu dinheiro. Assim, resolveu que irá comprar apenas uma colônia de bactérias. No entanto, ele pretende comprar a colônia que forneça a maior quantidade de bactérias ao final do período de crescimento da mesma.

Ele tem um catálogo mostrando os tipos de bactérias à venda. Para cada tipo de bactéria, o catálogo informa a quantidade de bactérias geradas por uma bactéria desse tipo a cada divisão e por quantos dias a população da colônia crescerá. Porém, a calculadora que ele tem em casa não é suficiente para que ele faça os cálculos necessários para decidir qual é a melhor colônia a comprar.

Tarefa

Bruno pediu sua ajuda para decidir qual é o melhor tipo de bactéria para a compra. Lembre que para Bruno o melhor tipo de bactéria é aquele cuja colônia, ao final do período de crescimento, terá a maior quantidade de bactérias.

Você deve supor que não haverá duas colônias com a mesma população final de bactérias.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado).

A primeira linha da entrada contém um inteiro N ($1 \leq N \leq 50.000$) representando a quantidade tipos de bactérias no catálogo. Cada uma das N linhas seguintes contém informações sobre um tipo de bactéria: a primeira dessas linhas contém a informação da bactéria de tipo 0, a segunda dessas linhas contém a informação sobre a bactéria de tipo 1, e assim por diante. A última dessas linhas contém a informação da bactéria de tipo $N - 1$.

A informação para cada tipo de bactéria é composta por dois números inteiros D e C ($1 \leq D \leq 2.000$ e $1 \leq C \leq 5.000$), onde D é quantidade de bactérias que cada bactéria deste tipo gera ao se dividir numa noite, e C é a quantidade de dias que a população de bactérias crescerá.

Saída

Seu programa deve imprimir, na *saída padrão*, um número inteiro entre 0 e $N - 1$ representando o tipo da bactéria que Bruno deverá comprar.

Informações sobre a pontuação

- Em um conjunto de casos de teste que totaliza 20 pontos, $N \leq 1.000$, $D^C \leq 1.000.000$.
- Em um conjunto de casos de teste que totaliza 80 pontos, $N \leq 2.000$, $D \leq 2.000$, $C \leq 5.000$.

Exemplos

Entrada	Saída
2 2 5 3 4	1

Entrada	Saída
5 2 1 4 5 30 4 20 6 2 154	4

Entrada	Saída
4 145 15 2 4999 3 3211 135 20	2

Temos que encontrar a maior potenciação (a^b) entre todos os valores de a e b dados.

Mas como comparar duas potenciações (a^b, c^d) se não podemos calculá-la por que podemos ter um overflow. Temos que utilizar um método para conseguir reduzir esses valores.

Vamos utilizar a função logarítmica que consegue transformar multiplicações em adições. Os logaritmos eram bastante úteis antes do advento das calculadoras eletrônicas. Mesmo nas calculadoras atuais a função logarítmica ainda é bastante útil por causa da sua

continuidade. Quando você faz 10^{π}, a calculadora usa a seguinte fórmula:

$e^{\pi \ln(10)}$

Mas voltando ao problema original como comparar duas potenciações (a^b, c^d).

$a^b > c^d \rightarrow \ln(a^b) > \ln(c^d) \rightarrow b * \ln(a) > d * \ln(c)$

```
#include <stdio.h>
#include <math.h>

int main() {
  int N, D, C, res, i;
  double bact = 0.0;

  scanf("%d", &N);
  for (i = 0; i < N; i++) {
    scanf("%d%d", &D, &C);
    if (bact < (double)C*log(D)) {
        bact = C*log(D);
      res = i;
    }
  }

  printf("%d\n", res);

  return 0;
}
```

Solução oficial dada pela organização da Olimpíada Brasileira de Informática

Mas vamos imaginar que não soubéssemos que existe a função log no arquivo de cabeçalho math.h[1]. E agora, José?[2]

Sabemos que a soma da série harmônica [3] 1,1/2,1/3,... diverge, ou seja,

$$\sum_{i=1}^{\infty} \frac{1}{i} = \infty$$

Mas para valores grandes de n, a soma dos n primeiros termos da série podem ser bem aproximados [4]

$$\sum_{i=1}^{n} \frac{1}{i} \approx \ln n + \gamma$$

```
#include <stdio.h>

#include <math.h>

double ln(int n){

      int i;

      double res = 0.0;

      for(i=1;i<=n;i++)

            res = res + 1.0/i;

      return res;

}

int main(){

      int n;

      while( scanf("%d",&n)  && n!=0 ){

            printf("%d %lf %lf\n",n,log(n) , ln(n) );

      }

}
```

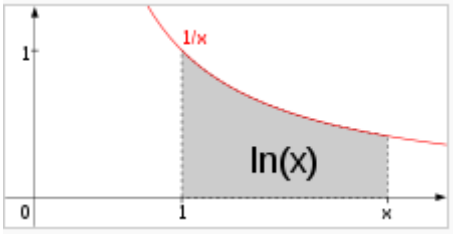
Saída:

```
100

100 4.605170 5.187378

1000

1000 6.907755 7.485471

10000

10000 9.210340 9.787606

100000

100000 11.512925 12.090146

1000000

1000000 13.815511 14.392727
```

Note que à medida que n vai crescendo a aproximação torna-se melhor. Mas vamos supor que você ainda não está ainda satisfeito e quer uma aproximação ainda melhor. Estamos podemos apelar para as aulas de cálculo I e para a seguinte fórmula.

$$\frac{d}{dx}\ln(x) = \frac{1}{x}, \qquad \int \frac{1}{x}\,dx = \ln(x) + C$$

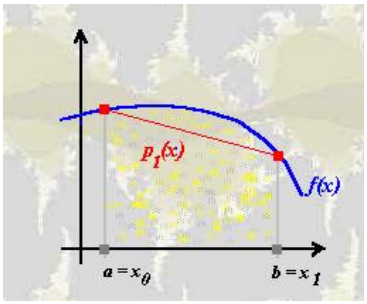
$$\ln(a) = \int_1^a \frac{1}{x}\,dx.$$



ln(x) pode ser definido como a área abaixo da curva da função f(x) = 1/x

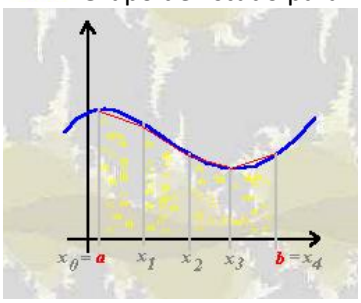
Então precisamos desenvolver um método para resolver esta integral por uma aproximação da área da área abaixo da curva. E agora, José?[2]

Podemos utilizar a regra do trapézio simples para tentar aproximar a área abaixo da curva.



$A = [f(x_0) + f(x_1)](b-a)/2$

Podemos decompor este intervalo em sub-intervalos cada vez mais pequenos , e nesses sub-intervalos podemos aplicar a regra do trapézio da maneira mais conveniente.



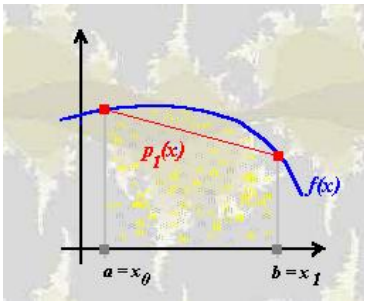
Podemos aproximar a região da curva decompondo a curva fixando um número de intervalos. Vamos considerar o número de intervalos igual ao número n.

```
#include <stdio.h>
#include <math.h>
#define eps 1.0e-6
double ln(int n,double intervalos){
	double fa =1, fb , area = 0.0;
	double largura = (n - 1) / intervalos;
	double x;
	x = 1.0;
	while ( x <= 1.0*n - eps ){
		x = x + largura;
		fb = 1.0/x;
		area = area + ((fa + fb)*(largura))/2.0;
		fa = fb;
	}
	return area;
}
int main(){
	int n;
	while( scanf("%d",&n)  && n!=0 ){
		printf("%d %lf %lf\n",n,log(n) , ln(n,n*1.0) );
	}
}
```

Saída:

```
100

100 4.605170 4.680934

1000

1000 6.907755 6.984826

10000

10000 9.210340 9.287542
```

Será que podemos fazer ainda melhor? Vou utilizar agora um método que não decompõe todos os intervalos do mesmo tamanho. Este método divide um intervalo da melhor maneira possível considerando uma tolerância eps. A escolha de dividir ou não um intervalo será baseado no valor da tolerância e na característica da curva. Este método é chamado *quadratura adaptativa*.



Primeiramente, vamos calcular o ponto médio m entre a e b. Então vamos aproximar a área de três regiões sobre a curva definida pela função f(): a para m, m para b e a para b. Se a soma das duas regiões menores está dentro da tolerância da maior área, então a aproximação é considerada boa o bastante. Se não, o problema inicial – região de a para b – é dividido em duas sub-regiões – de a para m e m para b – e o processo repete-se.

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define eps 1.0e-6
double f(double x){
	return 1.0/x;
}
double area(double a, double b){
	return (f(a) + f(b))*(b-a)/2.0;
}
double quad(double esq, double dir ){
	double m = (esq + dir)/2.0;
	double area_esq = area(esq,m);
	double area_dir = area(m,dir);
	double area_total = area(esq,dir);
	if(fabs( (area_esq + area_dir) - area_total) > eps ){
		area_esq = quad(esq,m);
		area_dir = quad(m,dir);
	}
	return (area_esq+area_dir);
}
int main(){
	int n;
	while( scanf("%d",&n)  && n!=0 ){
		printf("%d %lf %lf\n",n,log(n) , quad(1,n) );
	}
}
```

Saída:

```
10

10 2.302585 2.302606

100

100 4.605170 4.605212

1000

1000 6.907755 6.907817

10000

10000 9.210340 9.210424

100000

100000 11.512925 11.513029
```

"Qualquer tecnologia suficientemente avançada é indistinguível da magia"
Arthur C. Clarke

[1] http://pt.wikipedia.org/wiki/Math.h

[2] Carlos Drummond de Andrade, **Poema José**

[3] **http://matemateca.incubadora.fapesp.br/portal/matemateca/exposicao/harmonica/**

[4] **http://pt.wikipedia.org/wiki/S%C3%A9rie_harm%C3%B3nica_%28matem%C3%A1tica%29**

E agora, José?

A festa acabou,
a luz apagou,
o povo sumiu,
a noite esfriou,
e agora, José ?
e agora, você ?
você que é sem nome,
que zomba dos outros,
você que faz versos,
que ama protesta,
e agora, José ?

Está sem mulher,

está sem discurso,
está sem carinho,
já não pode beber,
já não pode fumar,
cuspir já não pode,
a noite esfriou,
o dia não veio,
o bonde não veio,
o riso não veio,
não veio a utopia
e tudo acabou
e tudo fugiu
e tudo mofou,
e agora, José ?

E agora, José ?
Sua doce palavra,
seu instante de febre,
sua gula e jejum,
sua biblioteca,
sua lavra de ouro,
seu terno de vidro,
sua incoerência,
seu ódio - e agora ?

Com a chave na mão
quer abrir a porta,
não existe porta;
quer morrer no mar,
mas o mar secou;
quer ir para Minas,
Minas não há mais.
José, e agora ?

Se você gritasse,
se você gemesse,
se você tocasse
a valsa vienense,
se você dormisse,
se você cansasse,
se você morresse…
Mas você não morre,
você é duro, José !

Sozinho no escuro
qual bicho-do-mato,
sem teogonia,
sem parede nua
para se encostar,
sem cavalo preto
que fuja a galope,
você marcha, José !
José, pra onde ?