Butters o Cracker

Mais uma vez Cartman vem com ideias mirabolantes para tentar mudar suas notas no sistema da escola de South Park. A ideia agora é atacar o gerador de números pseudo-aleatórios do sistema `mudou`, muito popular entre os professores desta pacata escola.

Claro que Cartman não sabe como fazer nada e –pediu– ordenou para que Butters fizesse o programa de computador que descobrisse a semente utilizada para a geração de números aleatórios.

Butters é muito jovem e um pouco esperto, e já descobriu algumas coisas interessantes.

O gerador de números pseudo-aleatórios do `mudou` é feito na linguagem `C` e usam a função pronta `rand_r()` que, por sua vez, recebe como argumento um ponteiro para um número inteiro que é chamado de `semente`.

A `semente` é extremamente fundamental para a segurança do sistema, pois é a partir dela que a ordem dos números pseudo-aleatórios é definida, ou seja, para uma mesma semente a ordem de números gerados pela função `rand_r()` será sempre a mesma.

Por exemplo:

Se você passar o número `380` como semente para a função `rand_r()` três vezes, a sequência de números geradas é: 633660840, 9717041, 491378313

Se o número da semente for `381`, os números gerados são:
1110404646, 1627573518, 1352223363

Butters descobriu que os números do `mudou` nunca ultrapassam 256, e por isso ele constatou que os números recebem um módulo 256. Logo a sequência dos números aleatórios para as sementes 380 e 381 são, respectivamente:

- 168 49 137
- 38 14 131

Você consegue implementar um programa muito simples que mostra isso como no exemplo abaixo:

```c
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
  int semente;
  printf("Digite a sua semente:\n");
```

```
7     scanf("%d",&semente);
8     for(int i=0;i<3;i++)
9       printf("%d\n",rand_r(&semente)%256);
10  }
```

- leia mais sobre a função `rand_r(3)` no manual.

Também foi descoberto pelo garoto prodígio, Butters, que a semente utilizada pelo `mudou` não é tão difícil de descobrir, eles utilizaram como semente o tempo em segundos desde 1 de janeiro de 1970 a partir do momento que a máquina foi ligada pela primeira vez. Logo, temos uma janela para descobrir qual é a possível semente.

Nesse ponto Butters está travado e pediu a sua ajuda para descobrir qual é a semente utilizada pelo sistema `mudou`.

Entrada

A entrada possui um único caso de teste. A primeira linha, do caso de teste, possui dois inteiros M_i e M_f (0 \leq M_i < M_f \leq 2^{31}) (cabe em um número inteiro sem sinal `int`), representando o intervalo das possibilidades da semente, sendo M_i o possível valor mais baixo e M_f o maior valor possível para a semente. Sabemos que a diferença entre M_f e M_i nunca é maior que 2^{17}.

A seguir, existem um conjunto de linhas, terminadas por `EOF`, indicando o qual número aleatório que o `mudou` gerou (em módulo 256) após 10000 gerações de números aleatórios.

Saída

A saída possui uma única linha contendo a semente utilizada pelo sistema `mudou`.

Exemplos

Exemplo de entrada

```
1 1
0
251
28
82
73
```

Exemplo de saída

```
1
```

Explicação para o caso de teste acima

A primeira linha é bastante simples e diz que o intervalo possível da semente varia de 1 até 1, notamos que a semente é claramente 1. A segunda linha é o valor gerador gerado pela chamada `rand_r(&semente)%256`, com a semente 1, depois de 10000 vezes, sendo ese o valor 0, a terceira linha representa o valor devolvido pela `rand_r` após mais 10000 execuções, devolvendo o valor 251. O código abaixo gera a mesma saída:

```
{
  int semente=1;
  for(int i=0;i<5;i++)
  {
    for(int j=1;j<10000;j++)
      rand_r(&semente);
    printf("%d\n",rand_r(&semente)%256);
  }
}
```

Exemplo de entrada

```
6 6
244
213
75
190
89
```

Exemplo de saída

```
6
```

Exemplo de entrada

```
921 936
141
156
139
126
84
174
238
53
99
26
```

Exemplo de saída

```
928
```

Explicação para o exemplo acima

No exemplo acima o intervalo é maior sendo a menor semente possível 921 e a maior semente possível 936. O seu programa precisa descobrir qual é a semente correta.

Para descobrir a melhor semente não há muito o que se fazer, você deverá simular a geração dos números aleatórios para cada uma das sementes e ir descartando quando descobrir que a sequência não é possível. No exemplo acima é fácil descartar todos, exceto o 928, pois apenas o 928 gera 141 nas primeiras 10000 iterações.

Exemplo de entrada

```
665 769
91
54
41
141
124
205
27
190
207
39
96
209
149
132
55
8
14
36
225
160
122
70
158
220
24
43
174
```

```
250
41
19
80
60
237
61
198
223
163
234
78
38
141
90
40
79
233
205
150
156
247
131
214
74
249
187
41
156
45
182
207
208
212
180
7
40
46
244
18
225
123
183
48
62
250
```

```
61
161
125
236
198
164
224
145
```

Exemplo de saída

```
730
```

Exemplo de entrada

```
34872413 34873777
92
155
191
116
38
4
250
181
163
239
135
25
16
155
166
222
176
73
151
69
194
57
153
142
133
171
237
248
58
222
```

```
211
197
32
19
138
51
121
138
84
130
131
131
111
244
127
61
28
199
173
249
154
60
76
247
42
147
157
119
12
11
224
184
128
229
85
251
198
97
59
128
29
191
211
135
198
63
```

```
93
79
```

Exemplo de saída

```
34872913
```

Explicação para o exemplo acima

Este já é um exemplo maior e mais elaborado, na primeira iteração você não pode descartar 6 possíveis sementes, pois elas geram o mesmo valor nas primeiras 10000 iterações, no caso são:
34872474, 34872584, 34872913, 34873242, 34873352, 34873681

Exemplo de entrada

```
16081291 16084449
43
48
145
154
89
155
236
153
177
255
16
50
114
156
62
165
221
180
182
49
50
133
184
24
177
80
132
151
```

```
153
85
89
241
43
212
120
100
167
12
32
50
76
62
147
153
92
170
15
217
21
143
213
52
184
46
37
232
132
199
62
54
187
154
97
93
155
231
206
159
101
237
197
58
88
237
```

```
134
111
181
39
80
125
188
218
100
166
173
72
1
40
200
175
105
68
76
79
218
57
```

Exemplo de saída

```
16082825
```

Author: Bruno Ribas