Número Proibido

Os números proibidos são números que possuem alguma representação problemática, por exemplo, número do azar, de algo ruim, e até números que são senhas do governo.

O número proibido mais conhecido é um número primo[1] que foi descoberto em 2001 e representa o arquivo binário da versão compactada do código C que implementa o algoritmo DeCSS, que pode ser utilizado para lograr o sistema de proteção do DVD.

Luan, um rapaz que tem muito receio de ser procurado por agências espiãs internacionais coletou um conjunto de números ilegais e está filtrando esses números de todos os seus arquivos no computador.

Infelizmente, Luan ainda não sabe programar muito bem e pediu a sua ajuda para implementar um programa que receba um conjunto de números ilegais e responda se um outro conjunto de números fazem parte dos números ilegais.

Entrada

A entrada é composta por um único caso teste que possui diversas linhas. A primeira linha possui um número N ($1 \leq N \leq 140000$) que representa a quantidade de números proibidos existentes. A segunda linha do caso de teste possui N números P_i ($0 \leq P_i \leq 2^{31}$) representando os números proibidos.

Depois existirão diversas linhas contendo um único número que se quer saber se é proibido ou não.

A entrada termina em EOF.

Saída

Para cada número da consulta deve-se imprimir uma única linha contendo a palavra `sim` se o número for proibido, ou `nao` caso o número não seja proibido.

Exemplo

Exemplo de entrada

```
7
10 0 50 25 121 0 3000
1
2
3
10
0
```

Saída para o exemplo de entrada acima

```
nao
nao
nao
sim
sim
```

Author: Bruno Ribas

[1] http://en.wikipedia.org/wiki/Illegal_prime

kk-página

Uma grande empresa de desenvolvimento de páginas de produtos eletrônicos está com um problema grave! Alguns *bots* e pessoas maliciosas conseguem deixar o servidor não responsivo com uma pesquisa bem curiosa.

Jaime, o rapaz do TI da empresa, percebeu que quando a pessoa clica no botão de consulta avançada do site e marca as opções: mostrar TODOS os produtos; ordenar por ID, e; ir para uma página X qualquer; o servidor demora a reponder (as vezes até minutos). Para piorar, se mais pessoas fazem isso, o servidor fica com várias consultas em execução e eventualmente para de responder totalmente.

O dono da empresa, *Istivi Trabalhos*, precisa de uma ajuda mais especializada e, não por acaso, te encontrou na lista de alunos de Engenharia de Software da UnB e gostou do seu perfil e requer a sua ajuda!

O problema já foi repassado para você e temos a parte que mais interessa.

> O seu programa será compilado com os parâmetros: `gcc -O2 -static arquivo.c -o arquivo`

Entrada

A entrada é composta por um único caso de teste contendo diversas linhas. A primeira linha, do caso de teste, possui três números inteiros:

- N ($0 <= N <= 2^{25}$), sendo a quantidade de produtos;
- P ($0 <= P <= 2^{16}$), sendo a página que deve ser apresentada;
- X ($1 <= X <= 100$), sendo a quantidade de produtos que aparecem por página;

A seguir são apresentada N linhas, cada uma contendo um inteiro ID_i ($0 <= ID_i <= 2^{31}$) representando o ID de um produto. Não existem ids repetidos.

Saída

Você deve imprimir os X IDs da página P, ordenados de forma não decrescente.

Exemplo

Exemplo de entrada

```
10 3 2
1
2
3
4
5
6
7
8
9
10
```

- Atenção: A página é indexada a partir de `0`, logo a página ($P =$)`3` representa a quarta página

Saída para o exemplo acima

```
7
8
```

Exemplo de entrada

```
10 1 3
248
125
378
268
343
45
78
71
297
150
```

Saída para o exemplo acima

```
125
150
248
```

Exemplo de entrada

```
9 4 2
106
210
270
67
69
127
303
236
249
```

Saída para o exemplo acima

```
303
```

- ATENÇÃO: Cuidado quando a impressão acontece na última página, podem sobrar menos elementos que o máximo para se mostrar em cada página

Author: Bruno Ribas

Analisando Dados de Degustação

Preâmbulo

A Empresa de Degustação Aguda (EDA) está criando uma bebida apurada e envelhecida nos melhores barris do mundo. Os barris são lavados e incinerados no espaço, o que prometem dar um sabor especial ao líquido armazenado.

A pesquisa e o desenvolvimento desta nova bebida gerou muita discórdia entre os pesquisadores e os famosos degustadores. A **EDA** está em um ponto em que precisa decidir entre os vários sabores criados, aquele que vai impactar o mercado da melhor maneira.

Para conseguir determinar o melhor sabor, a **EDA** contratou degustadores e amadores ao redor do mundo todo, e fez um experimento bastante curioso: A cada rodada de degustação a **EDA** pedia para que os degustadores experimentassem um conjunto de bebidas, classificadas por letras de `a` a `z` ou de `A` a `Z`, e depois disso eles pediam para que cada pessoa enviasse uma mensagem com a letra da bebida que eles mais gostaram. Dessa forma os dados recolhidos compõem uma *string* cheia de letrinhas. E os pesquisadores da **EDA** descobriram que a letra com a maior sequência contínua representa a melhor bebida.

Analisar os dados é um tanto quanto complexo e, por isso a **EDA** contratou VOCÊ para escrever um programa de computador que seja capaz de analisar o conjunto de dados coletados durante o experimento. Como as informações são sigilosas, você trabalhará com algumas informações *anonimizadas*.

A análise acontece da seguinte forma:

- Uma string com as letras das escolhas é passada para o seu programa;
- A posição em que cada uma começa é importante, a primeira começa na posição `0`;
- Você precisa contar o tamanho das sequências formadas pelo mesmo caractere, por exemplo:

  `aabbbcaaaa`

  - As sequências do exemplo acima são:
    - a começando na posição `0` composta por `2` ocorrências;
    - b começando na posição `2` composta por `3` ocorrências;
    - c começando na posição `5` composta por `1` ocorrência;
    - a começando na posição `6` composta por `4` ocorrencias
      - veja que contabilizamos as sequências com os mesmos caracteres independetemente.
- Após contar você deve apresentar os dados ordenados conforme a quantidade de ocorrências dos caracteres. Veja nas seções abaixo a explicação a respeito da entrada e saída de dados, bem como um conjunto limitado de exemplos.

Entrada

A entrada é composta por um único caso de teste. Cada caso de teste possui uma única linha contendo uma string `S` de comprimento `|S|`, sendo $1 \leq |S| \leq 100000$.

A string |S| contém qualquer conjunto de caracteres entre `[a-zA-Z]`, ou seja, qualquer caractere de `A` até `Z` sendo os minúsculos considerados diferentes dos maiúsculos. A string não possui caracteres de espaço e termina com `EOS` (*End Of String*), sendo representado pelo caractere de quebra de linha `"\n"`.

Saída

A saída é composta por diversas linhas. Cada linha deve conter três dados, são eles: um inteiro `I`; um caractere `C`, e; um inteiro `P`; representando respectivamente o tamanho da sequência; o caractere da sequência, e; a posição que o caractere começou na string `S` original.

A saída deverá estar ordenada de maneira não decrescente pelo indexador I e em caso de empate considere a sequência que apareceu antes na entrada.

Exemplos

Exemplo de entrada

```
aabbbcaaaa
```

Saída para o exemplo acima

```
4 a 6
3 b 2
2 a 0
1 c 5
```

Exemplo de entrada

```
aabbzzkkll
```

Saída para o exemplo acima

```
2 a 0
2 b 2
2 z 4
2 k 6
2 l 8
```

Exemplo de entrada

```
zlkzzzzzzzzzz
```

Saída para o exemplo acima

```
10 z 3
1 z 0
1 l 1
1 k 2
```

Exemplo de entrada

```
AAAfddafdadAffsaAfAssdasfaadaasAfafsfdaAAfAaaffAda
```

Saída para o exemplo acima

```
3 A 0
2 d 4
2 f 12
2 s 19
2 a 25
2 a 28
2 A 39
2 a 43
2 f 45
1 f 3
1 a 6
1 f 7
1 d 8
1 a 9
1 d 10
1 A 11
1 s 14
1 a 15
1 A 16
1 f 17
1 A 18
1 d 21
1 a 22
1 s 23
```

```
1 f 24
1 d 27
1 s 30
1 A 31
1 f 32
1 a 33
1 f 34
1 s 35
1 f 36
1 d 37
1 a 38
1 f 41
1 A 42
1 A 47
1 d 48
1 a 49
```

Exemplo de entrada

```
JdJGllflsAJalJjlajgJfajdjsdsjJJljsAaaaAGgfAsGlsagGfaGfgJaGsAAAddgGljfglfJjAlagJJassjgdsJfsfGgsAfJslg
```

Saída para o exemplo acima

```
3 a 35
3 A 59
2 l 4
2 J 29
2 d 62
2 J 78
2 s 81
1 J 0
1 d 1
1 J 2
1 G 3
1 f 6
1 l 7
1 s 8
1 A 9
1 J 10
1 a 11
1 l 12
1 J 13
1 j 14
1 l 15
1 a 16
1 j 17
1 g 18
1 J 19
1 f 20
1 a 21
1 j 22
1 d 23
1 j 24
1 s 25
1 d 26
1 s 27
1 j 28
1 l 31
1 j 32
1 s 33
1 A 34
1 A 38
1 G 39
1 g 40
1 f 41
```

```
1 A 42
1 s 43
1 G 44
1 l 45
1 s 46
1 a 47
1 g 48
1 G 49
1 f 50
1 a 51
1 G 52
1 f 53
1 g 54
1 J 55
1 a 56
1 G 57
1 s 58
1 g 64
1 G 65
1 l 66
1 j 67
1 f 68
1 g 69
1 l 70
1 f 71
1 J 72
1 j 73
1 A 74
1 l 75
1 a 76
1 g 77
1 a 80
1 j 83
1 g 84
1 d 85
1 s 86
1 J 87
1 f 88
1 s 89
1 f 90
1 G 91
1 g 92
1 s 93
1 A 94
1 f 95
1 J 96
1 s 97
1 l 98
1 g 99
```

Author: Bruno Ribas