OPEI 2016
Olimpíada Pernambucana de Informática

MODALIDADE D
PROVA PRÁTICA - ENSINO MÉDIO

Leia atentamente as seguintes Instruções.

- Esta prova tem início às 09:00 (nove) horas do dia 03 de Setembro de 2016.
- Esta prova, modalidade prática, possui duração de 3 (três) horas.
- O aluno só poderá deixar o local de prova a partir de 30 (trinta) minutos do horário inicial da prova.
- Os últimos 3 alunos restantes na sala, devem esperar até o final do tempo de prova para sair.
- Aguarde orientações quanto a submissão das respostas.
- Preencha à caneta os seus dados pessoais.
- É proibido realizar qualquer tipo de pesquisa ou consulta.

Nome completo: _____

Idade: ___ Ano escolar do aluno: ___ Nº do RG ou CPF: _____

Escola: _____

Organização

Centro de Informática UFPE

pet | Informática

Apoio

inlocomedia.com

***A solução da prova se encontra no fim da mesma.**

1) Conversão de hora

Tom conhece dois tipos de relógio, os relógios que mostram a hora de 0 até 23, e os relógios que mostram as horas de 1 até 12. Porém, ele tem dificuldades de ver as horas em relógios que mostram de 0 até 23, e por isso quer usa ajuda. Dado uma hora de 0 até 23, qual o respectivo horário no outro relógio?

Entrada:

A única linha da entrada contém um inteiro **H** indicando a hora mostrada no relógio.

Saída:

Um inteiro indicando qual hora seria mostrada no outro relógio.

Limites:

0 ≤ **H** ≤ 23

Exemplos:

Entrada:	Saída:		Entrada:	Saída:
0	12		12	12
Entrada:	Saída:		Entrada:	Saída:
5	5		17	5
Entrada:	Saída:			
23	11			

2) Os sete reinos

Nos sete reinos, quando uma pessoa é suspeita de um crime ela pode escolher ser julgada de várias maneiras diferentes, sendo a mais escolhida o julgamento por combate, onde a pessoa sendo julgada escolhe alguém para lutar por ela em uma luta que normalmente só acaba quando um dos dois lutadores morrem. Após algum tempo esse tipo de julgamento foi banido, e foi criado o julgamento por vôlei, onde ao invés da pessoa sendo julgada escolher alguém para lutar por ela até a morte, escolhe 6 pessoas para um time de vôlei que irá representá-la no julgamento. O julgamento por vôlei consiste em um

jogo de vôlei, em que caso a equipe da pessoa julgada ganhe ela é considerada inocente e caso perca, é considerada culpada do crime.

O jogo consiste de *sets*, a equipe que ganhar 3 *sets* primeiro é considerada a vencedora. Um *set* consiste de 25 pontos, a não ser que esse seja o quinto *set* do jogo, que consiste de apenas 15 pontos. Uma outra exceção é caso as duas equipes empatem com um ponto a menos do *set* acabar, i.e., as duas equipes fazem 24 pontos (ou 14 pontos caso seja o quinto *set*), nesse caso, a equipe considerada vencedora do *set* é a equipe que conseguir 2 pontos a mais que a outra equipe. Ou seja, em teoria, pode ser que um *set* nunca acabe caso as duas equipes fiquem alternando pontos.

O julgamento por vôlei passou a ser o modo favorito de julgamento, com isso, muitos jogos estão acontecendo e começou a ficar difícil de saber se as pessoas são inocentes ou culpadas. A sua tarefa é dado o estado atual do jogo dizer se a pessoa é inocente, culpada ou se o jogo ainda não acabou.

Entrada:

A primeira linha da entrada contém um inteiro **N**, o número de sets iniciados até o momento. Cada uma das **N** linhas seguintes contém dois inteiros **A B**, onde, na i-ésima linha, **A** indica a pontuação feita pelo time da pessoa sendo julgada no i-ésimo set da partida e **B** indica a pontuação feita pelo time oposto.

É garantido que todos os *sets* acabaram respeitando as regras acima, com exceção do último *set* que pode não ter acabado e ainda estar em andamento.

Saída:

A saída deve conter uma única frase (aspas para claridade):

- “inocente”, caso o time do réu tenha ganho
- “culpada”, caso o time do réu tenha perdido
- “em julgamento”, caso o jogo ainda não tenha acabado

Limites:

$1 \leq \mathbf{N} \leq 5$

$0 \leq \mathbf{A}, \mathbf{B} \leq 10^9$

Exemplos:

Entrada:	Saída:		Entrada:	Saída:
4 25 10 25 23 23 25 25 20	inocente		3 25 18 25 22 25 24	em julgamento

Entrada:	Saída:		Entrada:	Saída:
5 25 10 25 23 23 25 15 25 13 15	culpada		5 25 10 25 23 30 32 20 25 18 19	em julgamento
Entrada:	Saída:			
1 0 0	em julgamento			

3) Exercícios

João das Neves começou a praticar exercícios, e para continuar incentivado ele criou um programa que verifica seu batimento cardíaco durante os exercícios e atribui pontuações. João definiu três níveis de batimento cardíaco:

1. Abaixo da média, para cada minuto se exercitando com o batimento abaixo da média ele ganha **X** pontos.
2. Na média, para cada minuto se exercitando com o batimento na média ele ganha **Y** pontos.
3. Acima da média, para cada minuto se exercitando com o batimento acima da média ele ganha **Z** pontos.

João sabe que é impossível passar direto do nível 1 (abaixo da média) para o nível 3 (acima da média), assim como é impossível começar os exercícios no nível 2 (na média) ou 3. Ou seja, sempre que João das Neves se exercita, ele começa no nível 1, após alguns minutos entra no nível 2 e depois de mais alguns minutos ele entra no nível 3. Obviamente João pode parar o exercício a qualquer momento, ou seja, dependendo de quando ele pare é possível que nunca entre no nível 2 ou no 3. João sempre passa uma quantidade inteira de minutos em cada nível e nunca vai de um nível maior para um nível menor.

O problema é que o programa de João só marca que ele passou **N** minutos se exercitando no total e que ganhou **M** pontos no total, se João quisesse saber quantos minutos ele passou se exercitando em cada nível de batimento cardíaco ele poderia mudar o programa um pouco. Porém, uma informação que João não sabe como calcular rapidamente e gostaria de ajuda é em saber quantos jeitos válidos diferentes (**A**, **B**, **C**) existem, tal que se João passar **N** minutos se exercitando, onde **A** minutos são se exercitando no nível 1, **B** minutos no nível 2 e **C** minutos no nível 3, ele consiga exatamente **M** pontos.

Entrada:

A primeira linha da entrada contém **N** e **M**, a quantidade de minutos que João passou se exercitando e quantos pontos ele ganhou, respectivamente. A segunda linha da entrada contém **X**, **Y** e **Z**, quantos pontos ele ganha por minuto em cada nível de batimento cardíaco, como explicado acima.

Saída:

A saída contém apenas um número inteiro, o número de jeitos válidos distintos (**A**, **B**, **C**).

Limites:

$1 \leq \mathbf{N} \leq 10^6$
$1 \leq \mathbf{M} \leq 10^9$
$1 \leq \mathbf{X}, \mathbf{Y}, \mathbf{Z} \leq 10^3$ (**X** ≠ **Y** e **X** ≠ **Z** e **Y** ≠ **Z**)

Informações sobre a pontuação

Para um conjunto de casos de teste valendo pontuação parcial, $1 \leq \mathbf{N} \leq 100$

Exemplos:

Entrada:	Saída:
10 29 2 3 5	3

As três soluções possíveis são (3, 6, 1), (5, 3, 2) e (1, 9, 0), pois as três duram 10 minutos e produzem 29 pontos.

Entrada:	Saída:
20 78 2 4 6	10

As dez soluções possíveis são (2, 17, 1), (7, 7, 6), (1, 19, 0), (3, 15, 2), (8, 5, 7), (4, 13, 3), (5, 11, 4), (6, 9, 5), (9, 3, 8) e (10, 1, 9).

Entrada:	Saída:		Entrada:	Saída:
100 1096 15 7 11	49		999088 517747550 285 295 681	2128

4) Análises

Sansa gosta muito de acompanhar esportes, de diversas categorias, como vôlei, futebol americano, beisebol e *League of Legends*. Após cada partida de seus times preferidos Sansa anota o resultado (vitória ou derrota). Recentemente ela começou a pesquisar sobre a área de *Sport Analytics*, e decidiu utilizar os dados coletados para analisar o desempenho recente dos seus times. Para as últimas **X** partidas de seus times preferidos, ela deseja saber quantas foram vitórias, e quantas foram derrotas. Sansa ficou muito entediada em realizar essa tarefa manualmente, e para isso deseja sua ajuda para realizar rapidamente a tarefa e poder fazer mais análises dos seus times preferidos.

Time	Vitórias	Derrotas	Últimos 10 jogos
Boston Red Sox	70	54	8-2
Toronto Blue Jays	70	54	6-4
Baltimore Orioles	68	56	4-6
New York Yankees	63	61	5-5
Tampa Bay Rays	62	71	6-4

Entrada:

A primeira linha da entrada contém dois inteiros **N** e **X**, representando o número de eventos na entrada, e o número de partidas que devem ser consideradas. As **N** linhas seguintes contém eventos, que podem ser de 3 tipos: **V** indica uma vitória, **D** uma derrota e **P** uma pergunta de quantas vitórias e quantas derrotas ocorreram nos **X** últimos jogos ocorridos até então. Caso **X** seja maior do que o número de partidas ocorridas até o momento, considere todas partidas.

Saída:

Para cada evento do tipo **P X**, imprima uma linha no formato "**V**-**D**" (aspas para claridade), onde **V** indica o número de derrotas e **D** indica o número de derrotas.

Limites:

$2 \leq \mathbf{N} \leq 10^6$
$1 \leq \mathbf{X} \leq \mathbf{N}$

Informações sobre a pontuação

Para um conjunto de casos de teste valendo pontuação parcial, $2 \leq \mathbf{N} \leq 100$

Exemplos:

Entrada:	Saída:		Entrada:	Saída:
12 3 V V D P D D P V D V V P	2-1 0-3 2-1		12 6 V V D P D D P V D V V P	2-1 2-3 3-3

5) Memória

Steven Rogers tem 96 anos de idade, com tantos anos de vida ele teve que criar várias senhas diferentes para todas as coisas que ele precisava, já que não é seguro repetir a mesma senha em dois lugares distintos. Com tantas senhas diferentes começou a ficar difícil de lembrar de todas e, por isso, Rogers escreveu todas as suas senhas em um caderno, o que não é seguro! Para não deixar suas senhas tão desprotegidas, ele as escreveu de forma criptografada, ou seja, de uma forma que uma pessoa ao ler não soubesse instantaneamente como era a senha, além disso, Rogers entregou o caderno para você proteger enquanto ele combate a organização Hidra, para não correr o risco dela obter o caderno.

Rogers conseguiu se infiltrar na Hidra nesse exato momento e só precisa de uma determinada senha para conseguir derrotá-la de vez. Só tem um problema, ele não lembra como é a senha e só você pode ajudá-lo. Incrivelmente, ele lembra de todas as senhas que estão escritas no caderno, menos a senha que ele precisa saber agora!

As senhas estão escritas no caderno da seguinte maneira: para cada senha que Rogers quer escrever no caderno, ele primeiro a divide em bloquinhos contíguos não-vazios de letras, a condição de divisão só ele sabe. Após isso, ele escreve cada bloquinho da palavra em qualquer parte do caderno, não importando se a ordem relativa dos bloquinhos continua a mesma ou não. Ao fazer isso para todas as senhas que ele quer, fica quase impossível uma pessoa que não conhece as senhas de adivinhar todas sem ser testando todas as possibilidades possíveis.

Para você ajudar Rogers, ele irá dizer todas as senhas que ele lembra, com isso, você pode riscar os bloquinhos de cada uma das senhas do caderno e só restará os bloquinhos pertencentes à senha que ele não lembra e assim ele conseguirá lembrar qual a senha esquecida. Para agilizar o processo, ao invés de Rogers dizer cada senha, ele já vai dizer os bloquinhos daquela senha, já que ele não quer revelar como é feita a divisão da senha em bloquinhos.

Entrada:

A primeira linha da entrada contém um inteiro **N**, o número de bloquinhos escritos no caderno. Cada uma das próximas **N** linhas contém uma palavra não-vazia representando um bloquinho que está escrito no caderno. A linha seguinte contém um inteiro **M**, o número de senhas que Rogers lembra, i.e. existem **M** + 1 senhas no caderno, e cada uma das **M** linhas seguintes contém a descrição de cada uma das senhas que Rogers lembra. A descrição da i-ésima senha consiste de um inteiro $\mathbf{K_i}$ indicando quantos bloquinhos a senha possui, seguido de $\mathbf{K_i}$ palavras separadas por espaço, representando os bloquinhos da senha.

Saída:

Todos os bloquinhos pertencentes à senha esquecida separados por um espaço, ordenados lexicograficamente.

Limites:

$1 \leq \mathbf{N} \leq 300$

$0 \leq \mathbf{M} \leq 49$

$\sum_{i=1}^{M} K_i < N$

$K_i > 0$

O tamanho de cada bloco não excede 5.

Exemplos:

Entrada:	Saída:		Entrada:	Saída:
9 d ba da bc da da e tata a 2 2 ba tata 4 a bc d e	da da da		6 d ba bc e tata a 1 2 ba tata	a bc d e

6) Arya e os patinhos

Em uma tarde de domingo, Arya e seus amigos estavam sem nada para fazer e resolveram brincar de patinhos na lagoa. A brincadeira funciona da seguinte forma, Arya e seus **N** amigos sentam em um círculo e reversam turnos, no seu turno uma pessoa deve dizer a palavra atual e passar a vez para o próximo do círculo, o jogo segue até que um jogador diga a palavra atual errada.

A palavra atual é decidida da seguinte forma: primeiramente a frase a ser dita pelos jogadores é “1 patinho na lagoa”, ou seja, a primeira palavra é “1”, depois “patinho”, depois “na” e depois “lagoa”, isso quer dizer que no primeiro turno, o jogador tem que dizer “1”, no segundo turno tem que dizer “patinho”, no terceiro turno tem que dizer “na” e no quarto turno tem que dizer “lagoa”. Após isso, a frase a ser dita se transforma em “2 2 patinhos patinhos na na lagoa lagoa”, a próxima frase a ser dita é “3 3 3 patinhos patinhos patinhos na na na lagoa lagoa lagoa”. Ou seja, a **i**-ésima frase consiste do número **i** repetido **i** vezes, seguido da palavra “patinhos” (com exceção da primeira frase que é “patinho” no singular) repetida **i** vezes, seguida da palavra “na” repetida **i** vezes, seguida da palavra “lagoa” repetida **i** vezes.

Ou seja, as palavras que devem ser ditas são: 1, patinho, na, lagoa, 2, 2, patinhos, patinhos, na, na, lagoa, lagoa, 3, 3, 3, patinhos, patinhos, patinhos, na, na, na, lagoa, lagoa, lagoa, 4, 4, 4, 4, patinhos, patinhos, patinhos, patinhos, na, na, na, na, lagoa, lagoa, lagoa, lagoa, 5, 5, 5, 5, 5… e a cada palavra dita o próximo jogador do círculo tem que dizer a palavra atual.

O primeiro jogador a falar é o de índice 1, depois o de índice 2, até chegar no de índice **N** + 1, e continua repetindo (1, 2, …, **N** + 1, 1, 2, …, **N** + 1,...) até alguém errar.

Como o jogo acontece muito rápido fica difícil de saber se as pessoas estão falando as palavras certas, você tem que fazer um programa que analisa as palavras ditas e verifica se alguém errou.

Entrada:

A primeira linha da entrada contém dois inteiros **N** e **M**, indicando a quantidade de amigos de Arya que estão jogando com ela e quantos turnos ocorreram até o momento. Cada uma das próximas **M** linhas contém uma palavra, indicando que palavra o jogador daquele turno disse.

Saída:

Um inteiro representando o índice do primeiro jogador que falou a palavra errada e perdeu a rodada, ou -1 caso ninguém tenha errado.

Limites:

$1 \leq \mathbf{N} \leq 15$

$1 \leq \mathbf{M} \leq 300$

Nenhuma palavra excede 9 caracteres e contém apenas letras minúsculas a-z.

Exemplos:

Entrada:	Saída:
2 4 1 patinho na lagua	1

Existem 3 jogadores (Arya e mais 2), o jogador de índice 1 deveria ter dito lagoa e não lagua.

Entrada:	Saída:
1 1 um	1

O jogador 1 deveria ter dito 1 em vez de um.

Entrada:	Saída:
1 3 1 patinho na	-1

Ninguém errou até o momento.

Entrada:	Saída:
3 17 1 patinho na lagoa 2 2 patinhos patinhos na na lagoa lagoa 3 3 3 jujuba batata	4

Existem 4 jogadores (Arya e mais 3), o jogador 4 deveria ter dito patinhos ao invés de jujuba no penúltimo turno da entrada, e como o jogador 4 já errou não faz mal o jogador 1 ter dito batata errado.

Soluções

1) Conversão de hora

```
#include <bits/stdc++.h>
using namespace std;

int main() {
  int H; scanf("%d", &H);
  if (H == 0) H = 12;
  else if (H > 12) H -= 12;
  printf("%d\n", H);
```

```
  return 0;
}
```

2) Os sete reinos

```
#include <bits/stdc++.h>

using namespace std;

int main() {
  int n; scanf("%d", &n);
  assert(1 <= n && n <= 5);

  int a = 0, b = 0;
  bool not_over = false;

  for (int i = 1; i <= n; ++i) {
    int x, y; scanf("%d %d", &x, &y);
    assert(0 <= x && x <= 1000000000);
    assert(0 <= y && y <= 1000000000);

    int total;
    if (i != 5) total = 25;
    else total = 15;

    if (x >= total && x >= y + 2) ++a;
    else if (y >= total && y >= x + 2) ++b;
    else { assert(i == n); not_over = true; }
  }

  assert(a < 4 && b < 4);

  if (not_over || (a < 3 && b < 3) || a == b) puts("em julgamento");
  else if (a > b) puts("inocente");
  else if (b > a) puts("culpada");
  return 0;
}
```

3) Exercícios

```
#include <bits/stdc++.h>
using namespace std;

int main() {
  int N; long long M; scanf("%d %lld", &N, &M);
  long long X, Y, Z; scanf("%lld %lld %lld", &X, &Y, &Z);

  long long cnt = 0;
```

```
  for (long long b = 0; b <= N; ++b) {
    long long num = (M - b * Y - N * Z + b * Z);
    long long den = X - Z;

    if (num % den != 0) continue;
    long long a = num / den;
    long long c = N - a - b;
    if (a < 1 || a > N) continue;
    if (b < 0 || b > N) continue;
    if (c < 0 || c > N) continue;
    if (b == 0 && c != 0) continue;
    ++cnt;
  }

  cout << cnt << endl;
  return 0;
}
```

4) Análises

```
#include <bits/stdc++.h>
using namespace std;

char matches[1111111];

int main() {
  int N, X; scanf("%d %d", &N, &X);
  assert(N <= 1000000);
  assert(X <= N);

  int ptr = 0, v = 0, d = 0;
  for (int i = 0; i < N; ++i) {
    char op; scanf(" %c", &op);
    if (op == 'P') {
      printf("%d-%d\n", v, d);
    } else {
      if (ptr >= X) {
        if (matches[ptr - X] == 'V') --v;
        else --d;
      }
      matches[ptr++] = op;
      if (op == 'V') ++v;
      else ++d;
    }
  }
  return 0;
}
```

5) Memória

```
#include <bits/stdc++.h>
using namespace std;

char words[303][303];
bool used[303];

int main() {
  int N; scanf("%d", &N);
  for (int i = 0; i < N; ++i) {
    scanf("%s", words[i]);
    used[i] = 0;
  }

  int M; scanf("%d", &M);
  while (M--) {
    int K; scanf("%d", &K);
    while (K--) {
      char w[333]; scanf("%s", w);
      for (int i = 0; i < N; ++i) {
        if (!used[i] && strcmp(words[i], w) == 0) {
          used[i] = true;
          break;
        }
      }
    }
  }

  vector<string> ans;
  for (int i = 0; i < N; ++i) {
    if (!used[i]) {
      ans.push_back(string(words[i]));
    }
  }

  sort(ans.begin(), ans.end());
  bool st = true;
  for (string s : ans) {
    if (!st) printf(" ");
    st = false;
    cout << s;
  }
  cout << endl;
  return 0;
}
```

6) Arya e os patinhos

```
#include <bits/stdc++.h>
using namespace std;

vector<string> sentence = {
  "patinhos",
  "na",
  "lagoa"};


int main() {
  int N, M; scanf("%d %d", &N, &M);
  ++N;
  vector<string> words = {"1", "patinho", "na", "lagoa"};

  for (int i = 2; ; ++i) {
    for (int j = 0; j < i; ++j) {
      words.push_back(to_string(i));
    }

    for (string s : sentence) {
      for (int j = 0; j < i; ++j) {
        words.push_back(s);
      }
    }

    if ((int)words.size() > M) break;
  }

  for (int i = 0; i < M; ++i) {
    string s; cin >> s;
    if (s != words[i]) {
      cout << (i % N) + 1 << endl;
      return 0;
    }
  }

  cout << -1 << endl;
  return 0;
}
```