CADERNO DE PROBLEMAS

26/10/2022 – 14h às 17h30

Leia atentamente estas instruções antes de iniciar a resolução dos problemas. Este caderno é composto de 6 problemas, sendo que 2 deles estão descritos em inglês.

1. Não é permitido o uso de material impresso, livros, apostilas e dicionários. Apenas é permitido o uso de lápis, caneta, lapiseira, borracha, régua e papel para rascunho (o papel é fornecido pela comissão organizadora). O acesso à internet é bloqueado durante a realização da prova. A ajuda do ambiente de desenvolvimento como C, C++, Java ou Python pode ser utilizada.

2. A correção das resoluções dos problemas será automatizada, por meio do sistema de submissão eletrônica BOCA, tendo como base os resultados obtidos a partir de uma série de execuções dos algoritmos submetidos pelas equipes.

3. Siga atentamente as exigências da tarefa quanto ao formato da entrada e da saída do seu algoritmo. Não implemente nenhum recurso gráfico nas suas soluções (janelas, menus, etc.), nem utilize qualquer rotina para limpar a tela ou posicionar o cursor.

4. Os problemas não estão ordenados por ordem de dificuldade neste caderno. A sugestão é de que procure resolver primeiro os problemas mais fáceis.

5. Utilize os nomes indicados nos problemas para nomear os seus arquivos-fonte, de acordo com a linguagem de programação utilizada.

6. Os problemas devem ser resolvidos utilizando o raciocínio entrada-processamento-saída, ou seja, não é necessário armazenar toda a saída para depois exibi-la.

7. Dicas de leitura (entrada de dados) e exibições (saída de dados) encontram-se no verso desta folha.

Faculdade de Informática de Presidente Prudente

http://www.unoeste.br/fipp/maratona

Observações:

Não utilize arquivos para entrada ou saída. Todos os dados devem ser lidos da entrada padrão (teclado) e escritos na saída padrão (tela). Utilize as funções padrão para entrada e saída de dados, como os exemplos a seguir:

- em C: `scanf`, `getchar`, `printf`, `putchar`;
  Para ler um **char** use: `scanf(" %c",&c);` não use: `fflush(stdin); scanf("%c",&c);`

- em C++: as mesmas de C ou os objetos `cin` e `cout`;
  Em C ou C++: use `sprintf(str,"%d",num);` em vez de `itoa(num,str,10);`

  Em C ou C++: não use `include<conio.h>`

- em Java, pode-se usar a classe `Scanner`:

```
Scanner sc = new Scanner(System.in);
int i = sc.nextInt();
String s = sc.next();
float f = sc.nextFloat();
```

  Ou ainda as classes `InputStreamReader` e `BufferedReader`:

```
InputStreamReader isr = new InputStreamReader(System.in);
BufferedReader br = new BufferedReader(isr);
try {
    v = br.readLine();
} catch(Exception e){  }
```

  Para exibição:

```
System.out.println();
System.out.printf("%d", i); //também %f, %c, %s, ...
```

  Em Java: não utilize `package`, o arquivo fonte deve ter a classe com os métodos e o `main`.

- em Python:
  Para leitura de um **int** use: `i = int(input())`
  Para leitura de um **float** use: `f = float(input())`
  Para leitura de um **char** use: `c = input()[0]`
  Para leitura de valores sequencias (quantidade fixa): `10 20 30`

```
v1, v2, v3 = list(map(int, input().split()))
```

  Para leitura de valores sequencias (quantidade não fixa): `10 20 30 ....`

```
valores = list(map(int, input().split()))
for v in valores:
    print(v) # faz alguma coisa com os itens da lista
```

  Para leitura de várias strings (quantidade não fixa): `aaa bbbb ccc`

```
strings = list(map(str, input().split()))
for s in strings:
    print(s) # faz alguma coisa com as strings da lista
```

Jogo de Varetas (amarela)

Problema
A

Arquivo fonte: *varetas.c*, *varetas.cpp*, *varetas.java* ou *varetas.py*

Há muitos jogos divertidos que usam pequenas varetas coloridas. A variante usada neste problema envolve a construção de retângulos. O jogo consiste em, dado um conjunto de varetas de comprimentos variados, desenhar retângulos no chão, utilizando as varetas como lados dos retângulos, sendo que cada vareta pode ser utilizada em apenas um retângulo, e cada lado de um retângulo é formado por uma única vareta. Nesse jogo, duas crianças recebem dois conjuntos iguais de varetas. Ganha o jogo a criança que desenhar o maior número de retângulos com o conjunto de varetas. Dado um conjunto de varetas de comprimentos inteiros, você deve escrever um programa para determinar o maior número de retângulos que é possível desenhar.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém um inteiro N que indica o número de diferentes comprimentos de varetas ($1 \leq N \leq 1.000$) no conjunto. Cada uma das N linhas seguintes contém dois números inteiros C_i e V_i, representando respectivamente um comprimento ($1 \leq C_i \leq 10.000$) e o número de varetas com esse comprimento ($1 \leq V_i \leq 1.000$). Cada comprimento de vareta aparece no máximo uma vez em um conjunto de teste (ou seja, os valores C_i são distintos). O final da entrada é indicado por $N = 0$.

A entrada deve ser lida da entrada padrão.

Saída

Para cada caso de teste da entrada seu programa deve produzir uma única linha na saída, contendo um número inteiro, indicando o número máximo de retângulos que podem ser formados com o conjunto de varetas dado.

A saída deve ser escrita na saída padrão.

Exemplo de entrada	**Saída para o exemplo de entrada**
1 10 7 4 50 2 40 2 30 4 60 4 5 15 3 6 3 12 3 70 5 71 1 0	1 3 2

Loteria (rosa)

Problema B

Arquivo fonte: *loteria.c*, *loteria.cpp*, *loteria.java* ou *loteria.py*

As chances de ganhar na loteria nacional são pequenas e, portanto, seus colegas de faculdade decidiram organizar uma loteria particular, com sorteios todas as sextas-feiras. A loteria é baseada em um estilo popular: um aluno que quer apostar escolhe C números distintos de 1 a K e paga US$ 1,00 (observe que loterias tradicionais como a US *National Lotto* usam $C = 6$ e $K = 49$). Na sexta-feira, durante o almoço, os números C (também de 1 a K) são sorteados. O aluno cuja aposta tiver o maior número de acertos recebe o valor arrecadado nas apostas. Esse valor é dividido em caso de empate e acumula para a próxima semana caso ninguém adivinhe nenhum dos números sorteados.

Alguns de seus colegas não acreditam nas leis da probabilidade e pediram que você escrevesse um programa que determinasse os números que foram sorteados menos vezes considerando todos os sorteios anteriores, para que pudessem apostar nesses números.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém três inteiros N, C e K que indicam respectivamente o número de sorteios que já aconteceram ($1 \leq N \leq 10000$), quantos números compõem uma aposta ($1 \leq C \leq 10$) e o número máximo dos números a serem escolhidos em uma aposta ($C < K \leq 100$). Cada uma das próximas N linhas contém C inteiros distintos X_i indicando os números sorteados em cada disputa anterior ($1 \leq X_i \leq K$, para $1 \leq i \leq C$). O fim da entrada é indicado por $N = C = K = 0$.

A entrada deve ser lida da entrada padrão.

Saída

Para cada caso de teste na entrada, seu programa deve escrever uma linha de saída, contendo o conjunto de números que foram sorteados o menor número de vezes. Este conjunto deve ser impresso como uma lista, em ordem crescente de números. Deixe um espaço em branco entre dois números consecutivos na lista.

A saída deve ser escrita na saída padrão.

Exemplo de entrada	Saída para o exemplo de entrada
5 4 6 6 2 3 4 3 4 6 5 2 3 6 5 4 5 2 6 2 3 6 4 4 3 4 3 2 1 2 1 4 4 3 2 1 4 3 0 0 0	1 1 2 3 4

Odd or Even (verde)

Problema C

Arquivo fonte: *odd.c*, *odd.cpp*, *odd.java* ou *odd.py*

There are several versions of Odd or Even, a game played by competitors to decide random issues (such as "who will code this problem?"). In one of the versions, for two players, the game starts with each player calling either odds or evens. Then they count to three (some people chant, "Once, twice, three, SHOOT!"). On three, both players hold out one of their hands, showing a number of fingers (from zero to five). If the fingers add to an even number, then the person who called evens wins. If the fingers add to an odd number, then the person who called odds wins.

John and Mary played several games of Odd or Even. In every game, John chose odds (and, consequently, Mary chose evens). During the games each player wrote down, in small cards, how many fingers he/she showed, using one card for each game – Mary used blue cards, John used red cards. Their objective was to be able to re-check the results later, looking at the cards for each game. However, at the end of the day, John dropped the deck of cards, and although they could separate the cards by color, they are now out of order.

Given the set of numbers written on red cards and on blue cards, you must write a program to determine the minimum number of games that Mary certainly won.

Input

The input contains several test cases. The first line of a test case contains an integer N representing the number of games played ($1 \leq N \leq 100$). The second line of a test case contains N integers X_i, indicating the number of fingers shown by Mary in each of the games ($0 \leq X_i \leq 5$, for $1 \leq i \leq N$). The third line of a test case contains N integers Y_i, indicating the number of fingers shown by John in each of the games ($0 \leq Y_i \leq 5$, for $1 \leq i \leq$ N). The end of input is indicated by $N = 0$.

The input must be read from standard input.

Output

For each test case your program must write one line, containing one integer, indicating the minimum number of games that Mary certainly won.

The output must be written to standard output.

Sample input	**Output for the sample input**
3 1 0 4 3 1 2 9 0 2 2 4 2 1 2 0 4 1 2 3 4 5 0 1 2 3 0	0 3

Feynman (vermelha)

Problema D

Arquivo fonte: *feynman.c, feynman.cpp, feynman.java* ou *feynman.py*

Richard Phillips Feynman was a well-known American physicist and a recipient of the Nobel Prize in Physics. He worked in theoretical physics and also pioneered the field of quantum computing. He visited South America for ten months, giving lectures and enjoying life in the tropics. He is also known for his books “Surely You’re Joking, Mr. Feynman!” and “What Do You Care What Other People Think?”, which include some of his adventures below the equator.

His life-long addiction was solving and making puzzles, locks, and cyphers. Recently, an old farmer in South America, who was a host to the young physicist in 1949, found some papers and notes that is believed to have belonged to Feynman. Among notes about mesons and electromagnetism, there was a napkin where he wrote a simple puzzle: “how many different squares are there in a grid of N × N squares?”.

In the same napkin there was a drawing which is reproduced below, showing that, for N = 2, the answer is 5.

Input

The input contains several test cases. Each test case is composed of a single line, containing only one integer *N*, representing the number of squares in each side of the grid ($1 \leq N \leq 100$). The end of input is indicated by a line containing only one zero.

The input must be read from standard input.

Output

For each test case in the input, your program must print a single line, containing the number of different squares for the corresponding input.

The output must be written to standard output.

Sample input	**Output for the sample input**
2 1 8 0	5 1 204

Desempilhando Caixas (azul)

Problema E

Arquivo fonte: *caixas.c*, *caixas.cpp*, *caixas.java* ou *caixas.py*

Joãozinho e sua família acabaram de se mudar. Antes da mudança, ele colocou todos os seus livros dentro de várias caixas numeradas. Para facilitar a retirada dos livros, ele fez um inventário, indicando em qual caixa cada livro foi colocado, e o guardou na caixa de número 1.

Chegando no seu novo quarto, ele viu que seus pais guardaram as caixas em várias pilhas, arrumadas em fila, com cada pilha encostada na pilha seguinte. Joãozinho é um garoto muito sistemático; por isso, antes de abrir qualquer outra caixa, ele quer recuperar seu inventário.

Joãozinho também é um garoto muito desajeitado; para tirar uma caixa de uma pilha, ele precisa que a caixa esteja no topo da pilha e que ao menos um de seus lados, não importa qual, esteja livre (isto é, não tenham nenhuma caixa adjacente). Para isso, Joãozinho precisa desempilhar algumas das caixas. Como o quarto dele é bem grande, ele sempre tem espaço para colocar as caixas retiradas em outro lugar, sem mexer nas pilhas que os pais dele montaram.

Para minimizar seu esforço, Joãozinho quer que você escreva um programa que, dadas as posições das caixas nas pilhas, determine quantas caixas Joãozinho precisa desempilhar para recuperar seu inventário.

Entrada

A entrada é composta de vários casos de teste. A primeira linha de cada caso de teste contém dois números inteiros N e P, indicando, respectivamente, o número de caixas e o número de pilhas ($1 \leq P \leq N \leq 1.000$). As caixas são numeradas sequencialmente de 1 a N.

Cada uma das P linhas seguintes descreve uma pilha. Cada linha contém um inteiro Q_i, indicando quantas caixas há na pilha i, seguido de um espaço em branco, seguido de uma lista de Q_i números, que são os identificadores das caixas. Os elementos da lista são separados por um espaço em branco.

Todas as pilhas contêm pelo menos uma caixa, e todas as caixas aparecem exatamente uma vez na entrada. As caixas em cada pilha são listadas em ordem, da base até o topo da pilha. Todas as caixas têm o mesmo formato.

O final da entrada é indicado por $N = P = 0$.

A entrada deve ser lida da entrada padrão.

Saída

Para cada caso de teste da entrada, seu programa deve imprimir uma única linha, contendo um único inteiro: o número mínimo de caixas, além da caixa 1, que Joãozinho precisa desempilhar para recuperar o seu inventário.

A saída deve ser escrita na saída padrão.

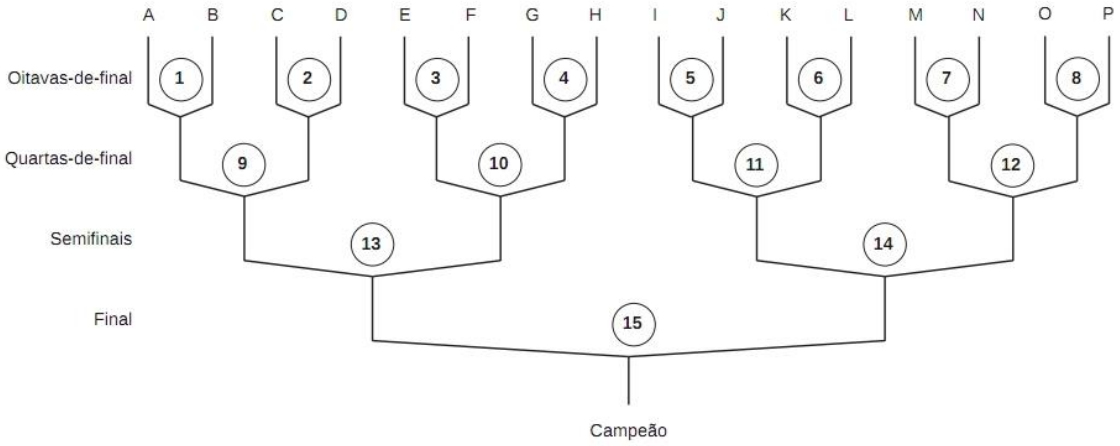
Exemplo de entrada	Saída para o exemplo de entrada
4 3 1 3 2 1 2 1 4 4 3 1 3 2 2 1 1 4 0 0	2 0

Copa do mundo (laranja)

Problema F

Arquivo fonte: *copa.c*, *copa.cpp*, *copa.java* ou *copa.py*

Este ano tem Copa do Mundo! O país inteiro se prepara para torcer para a equipe canarinho conquistar mais um título, tornando-se hexacampeã. Na Copa do Mundo, depois de uma fase de grupos, dezesseis equipes disputam a Fase final, composta de quinze jogos eliminatórios. A figura abaixo mostra a tabela de jogos da Fase final:



Na tabela de jogos, as dezesseis equipes finalistas são representadas por letras maiúsculas (de A a P), e os jogos são numerados de 1 a 15. Por exemplo, o jogo 3 é entre as equipes identificadas por E e F; o vencedor desse jogo enfrentará o vencedor do jogo 4, e o perdedor será eliminado. A equipe que vencer os quatro jogos da Fase final será a campeã (por exemplo, para a equipe K ser campeã ela deve vencer os jogos 6, 11, 14 e 15.

Dados os resultados dos quinze jogos da Fase final, escreva um programa que determine a equipe campeã.

Entrada

Cada entrada é composta de quinze linhas, cada uma contendo o resultado de um jogo. A primeira linha contém o resultado do jogo de número 1, a segunda linha o resultado do jogo de número 2, e assim por diante. O resultado de um jogo é representado por dois números inteiros M e N separados por um espaço em branco, indicando respectivamente o número de gols da equipe representada à esquerda e à direita na tabela de jogos ($0 \leq M \leq 20$, $0 \leq N \leq 20$ e $M \neq N$). O final da entrada é indicado por $N = 0$ e $M = 0$.

A entrada deve ser lida da entrada padrão.

Saída

Seu programa deve imprimir uma única linha, contendo a letra identificadora da equipe campeã.

A saída deve ser escrita na saída padrão.

Exemplo de entrada	**Saída para o exemplo de entrada**
4 1 1 0 0 4 3 1 2 3 1 2 2 0 0 2 1 2 4 3 0 1 3 2 3 4 1 4 1 0 2 0 1 0 2 1 1 0 1 0 1 2 1 2 1 0 2 1 1 0 0 1 0 2 2 1 1 0 2 1 0 0	F A