Universidade Federal de Goiás – UFG
Instituto de Informática – INF
Bacharelados (Núcleo Básico Comum)

Algoritmos e Estruturas de Dados 1 – 2020/2

Lista de Exercícios nº 06 – Algoritmos de Ordenação Interna

(Turmas: INF0286/INF0061 – Prof. Wanderley de Souza Alencar)

Sumário

1 Organizador de Vagões

 (++)

Na estação de trem você ainda pode encontrar o último dos(as) *organizadores(as) de vagões* de trens (OVT).
Um(a) OVT é um(a) prestador(a) de serviços, terceirizado(a), cujo trabalho é apenas o de *reordenar* os vagões do trem, trocando-os de posição conforme determinado.
Uma vez que os vagões são organizados numa ordem considerada ótima, o condutor pode desconectar cada vagão e colocá-los na estação.
O título OVT é dado à pessoa que realiza esta tarefa, cuja estação fica perto de uma ponte. Ao invés da ponte poder subir ou descer, ela roda sobre um pilar que fica no centro do rio. Após rodar 90, os barcos podem passar pela lateral esquerda ou pela lateral direita dela.
O(A) primeiro(a) OVT descobriu que girando a ponte em 180 graus com dois vagões em cima dela, é possível a troca de lugar entre os dois vagões. Obviamente a ponte pode operar no máximo com dois vagões sobre ela.
Agora que quase todos os(as) OVTs já faleceram, a estação gostaria de automatizar esta operação. Parte do sistema a ser desenvolvido para realizar esta automatização é uma rotina que decide, para um dado trem com determinado número de vagões, qual é o número de trocas entre trens adjacentes que são necessárias para que o trem fique ordenado (em ordem crescente).
Você, sendo da equipe de desenvolvimento da tarefa, já sabe: você deve elaborar tal rotina.

Entrada

A entrada contém, na primeira linha, o número de caso de testes $n \in \mathbb{N}^*$, com $1 \leq n \leq 50$.
Cada par de linhas seguintes representa um caso de teste, sendo que a primeira linha do par contém um número natural L, determinando o tamanho do trem $(1 \leq L \leq 1000)$, ou seja, o número de vagões nele presente.
A segunda linha do par contém uma *permutação qualquer* dos naturais de 1 até L, indicando a ordem atual dos vagões daquele trem.
Os vagões devem ser ordenados em ordem crescente pelo número de cada vagão.

Saída

Deve ser impresso, por caso de teste, o número ideal de trocas necessário para realizar a tarefa de ordenação dos vagões do trem.

Exemplos

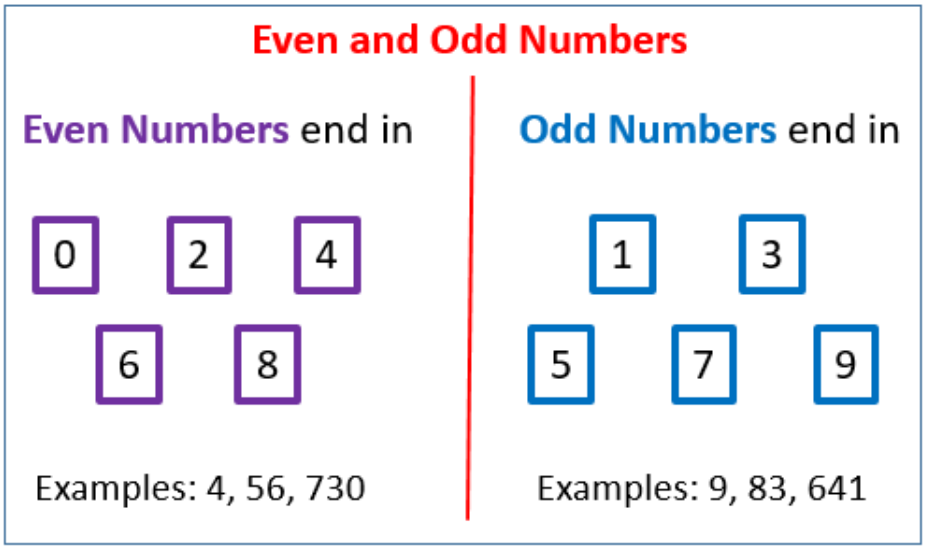
Entrada	Saída
3 3 1 3 2 4 4 3 2 1 2 2 1	1 6 1

Entrada	Saída
2 10 1 5 6 3 7 8 9 10 11 50 10 1 2 3 4 5 6 7 8 9 10	2 0

Entrada	Saída
3 10 1 5 2 6 3 7 4 8 9 10 1 3 10 10 9 8 7 6 5 4 3 2 1	6 0 45

Entrada	Saída
3 4 4 153 3 2 6 10 26 3 43 13 42 6 10 9 8 7 6 5	5 5 15

Entrada	Saída
3 8 1 2 3 5 7 4 9 8 1 4 20 1 2 3 4 5 6 7 8 9 10 11 12 13 15 17 18 19 20 14 16	3 0 9



2 Separando Números Pares de Ímpares

 (++)

Tales, um menino muito levado, pegou na escola uma caixa repleta de números naturais impressos em cartelas de EVA (*Espuma Vinílica Acetinada*) e derrubo-os sobre o chão da sala de aula.
Por estranho que pareça, ao cairem os números formaram uma *fila indiana* de tal maneira que ficaram com seus valores distribuídos aleatoriamente nesta fila.
Sabe-se que na caixa havia $n \in \mathbb{N}^*$ números, com $(1 < n \leq 100)$, mas seus valores são desconhecidos.
Sua tarefa é conceber um programa $\mathbb{C}$ que seja capaz de ordenar esta fila, segundo as seguintes regras:

- primeiro devem vir todos os números pares, em ordem crescente;
- depois devem vir os números ímpares, em ordem decrescente.

Entrada

A primeira linha de entrada contém o número n, quantidade de números existente na caixa que Tales derrubou.
A segunda linha contém os n números naturais, na ordem em que formaram a fila indiana, sempre separados por um único espaço em branco entre eles.

Saída

A saída deverá ter duas linhas. Na primeira são apresentados os números pares e na segunda os números ímpares, sempre separados por um único espaço em branco entre eles, conforme a ordem definida anteriormente.

Exemplos

Observação: Note que se, como exceção, a saída poderá ter uma única linha, se os números inicialmente fornecidos forem todos pares ou todos ímpares.

Entrada	Saída
10 4 32 34 543 3456 654 567 87 6789 98	4 32 34 98 654 3456 6789 567 543 87

Entrada	Saída
7 2 5 6 51 512 913 375	2 6 512 913 375 51 5

Entrada	Saída
8 6 2 8 12 202 304 18 10	2 6 8 10 12 18 202 304

Entrada	Saída
8 1 3 5 7 11 23 45 81	81 45 23 11 7 5 3

Entrada	Saída
5 10 8 6 4 2	2 4 6 8 10

3 Altura

 (+)

O governo brasileiro, sempre cheio de *boas ideias*, resolveu criar a "bolsa para desenolvimento da altura" (BDA), subsidiado na intenção de que as futuras gerações de brasileiros(as) sejam, em média, mais altas que a atual geração.
Você, neste momento, faz parte da equipe que realizará o levantamento da altura da população brasileira em todas as cidades do país. São mais de 5000 municípios, e ainda há projetos tramitando no Congresso Nacional para a criação de mais de 400 outros.
Na equipe, você foi designado para realizar a tarefa no conjunto de cidades com 10.000 ou menos habitantes. Nestas cidades terá que ser coletada a altura de *todos* os habitantes. Segundo o IBGE não há, atualmente no Brasil, nenhuma pessoa com mais que 230cm de altura.

Entrada

A primeira linha de entrada contém um número natural η_C, $\eta_C \leq 100$, que indica a quantidade de casos de teste, ou seja, o número de cidades que será a seguir informado.
Cada caso de teste é formado por duas linhas:
A primeira linha contém um número natural n, $1 \leq n \leq 10^4$, que indica a quantidade de pessoas daquela cidade.
A segunda linha apresenta a altura de cada uma destas pessoas, em centímetros, representado pela letra h, $20 \leq h \leq 230$, e separados por um espaço em branco entre si.

Saída

Deve-se imprimir, por caso de teste, uma linha contendo os valores das alturas de todos os moradores da cidade (em cm), em ordem crescente de altura, separados por um espaço em branco entre eles.

Entrada	Saída
3 20 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 20 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 100 42 184 82 33 70 126 21 214 149 93 71 158 79 224 188 120 200 81 154 215 162 164 100 67 103 123 51 56 202 162 24 96 99 124 51 36 69 43 51 114 102 152 133 72 43 159 58 229 39 52 64 82 115 172 138 33 108 111 192 131 170 50 61 195 103 164 184 51 51 69 141 144 84 140 38 99 228 35 199 35 164 30 77 163 23 119 104 189 104 121 160 24 140 135 35 68 199 151 178 87	33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 21 23 24 24 30 33 33 35 35 35 36 38 39 42 43 43 50 51 51 51 51 51 52 56 58 61 64 67 68 69 69 70 71 72 77 79 81 82 82 84 87 93 96 99 99 100 102 103 103 104 104 108 111 114 115 119 120 121 123 124 126 131 133 135 138 140 140 141 144 149 151 152 154 158 159 160 162 162 163 164 164 164 170 172 178 184 184 188 189 192 195 199 199 200 202 214 215 224 228 229

Exemplos

Observação

Devido à largura da coluna de **entrada**, algumas das sequências aparecerem em “duas linhas” mas, em verdade, serão digitadas numa única linha. No primeiro exemplo, a sequência: 45 186 185 55 51 51 22 78 64 26 49 21, deve ser digitada numa única linha.

Entrada	Saída
6 10 65 31 37 37 72 76 61 35 57 37 12 45 186 185 55 51 51 22 78 64 26 49 21 10 20 93 203 67 64 225 112 81 58 180 8 169 189 220 228 68 32 214 180 6 133 55 67 166 112 41 4 39 38 120 55	31 35 37 37 37 57 61 65 72 76 21 22 26 45 49 51 51 55 64 78 185 186 20 58 64 67 81 93 112 180 203 225 32 68 169 180 189 214 220 228 41 55 67 112 133 166 38 39 55 120

Entrada	Saída
5 4 66 66 86 67 8 185 188 189 190 191 192 193 194 7 201 93 203 32 173 230 27 10 55 120 22 112 81 90 33 118 80 160 3 51 20 70	66 66 67 86 185 188 189 190 191 192 193 194 27 32 93 173 201 203 230 22 33 55 80 81 90 112 118 120 160 20 51 70

Entrada	Saída
7 6 66 66 66 67 90 91 6 85 88 89 90 91 92 10 20 32 73 230 27 230 118 210 160 99 6 55 210 222 154 111 123 4 133 220 210 155 3 122 123 143 2 99 100	66 66 66 67 90 91 85 88 89 90 91 92 20 27 32 73 99 118 160 210 230 230 55 111 123 154 210 222 133 155 210 220 122 123 143 99 100

Entrada	Saída
5 6 66 66 66 67 90 91 5 85 88 89 90 91 10 20 32 73 230 27 230 118 210 160 99 6 55 210 222 154 111 123 4 133 220 210 155	66 66 66 67 90 91 85 88 89 90 91 20 27 32 73 99 118 160 210 230 230 55 111 123 154 210 222 133 155 210 220

4 PLACAR – Quem vai ser reprovado?

 (+++)

O professor Charles Francis Xavier, *Professor X*, aplicou para seus(suas) alunos(as) uma *Lista de Exercícios* contendo um conjunto de 10 (dez) "*problemas*" e concedeu o prazo de um mês para que eles(as) os resolvessem.
No final do mês os(as) alunos(as) enviaram, para o *Profesor X*, o número de problemas resolvidos corretamente.
A promessa do professor era reprovar, sumariamente, o(a) último(a) colocado(a) desta competição, onde os(as) alunos(as) seriam ordenados(as) conforme o número de problemas resolvidos, com empates resolvidos de acordo com a ordem alfabética dos nomes[1].
Este padrão fez com que alunos(as) com nomes iniciados nas últimas letras do alfabeto se esforçassem muito nas tarefas, e não compartilhassem suas soluções com colegas, especialmente aqueles(as) cujos nomes começassem com letras anteriores às deles(as).
Você é o(a) "*monitor(a)*" do *Professor X* e sua tarefa,é escrever um programa $\mathbb{C}$ que leia os resultados dos(as) alunos(as) e imprima a classificação final, marcando com a *hashtag* `#reprovado(a)` aquele(a) estudante que deverá ser reprovado(a).

Entrada

A primeira linha de cada contém um número natural n, $1 \leq n \leq 100$, que indica a quantidade estudantes na competiação.
Cada uma das n linhas seguintes contém o nome do(a) aluno(a) – sem espaço entre as suas palavras formadoras de seu nome – e o número de problemas resolvidos por ele(a). O número de problemas resolvidos está, obviamente, entre 0 e 10, inclusive extremos.

Saída

O programa de computador criado por você deverá imprimir o nome do(a) aluno(a) e o número de exercícios realizados, ordenadamente. Além disso, o(a) último(a) colocado deverá ser seguido pela *hashtag* `#reprovado(a)`.

[1] Suponha, para simplificação, que não há homônimos na turma e que todos os nomes dos(as) estudantes são escritos utilizando uma única palavra, sem espaços em branco entre elas, e que terá, no máximo, 20 símbolos. Por exemplo: AnaLuiza, Guilherme, Alice, LuizFelipe, Marcelo, PedroAugusto, etc.

Exemplos

Entrada	Saída
10 joaozinho 9 marcela 8 marcos 9 frodo 7 jailson 10 jolei 8 andre 10 maria 8 beatriz 10 ana 9	andre 10 beatriz 10 jailson 10 ana 9 joaozinho 9 marcos 9 jolei 8 marcela 8 maria 8 frodo 7 #reprovado(a)

Entrada	Saída
3 pedro 3 jose 3 arnaldo 3	arnaldo 3 jose 3 pedro 3 #reprovado(a)

Entrada	Saída
5 amanda 10 ananda 10 abadia 10 andreia 10 andubio 10	abadia 10 amanda 10 ananda 10 andreia 10 andubio 10 #reprovado(a)

Observação: Note que após o número do último(a) colocado(a) há um único espaço em branco antes da *hashtag* `#reprovado(a)` e não há espaço entre o último "o" e o símbolo de abre parêntese.

Entrada	Saída
4 cardonha 9 infelizreprovado 3 marcel 9 infelizaprovado 3	cardonha 9 marcel 9 infelizaprovado 3 infelizreprovado 3 #reprovado(a)

5 Insertion - Selection

 (+)

Escreva um programa ℂ que, a partir de um vetor de números naturais fornecido como entrada, calcule a diferença entre o número de trocas realizadas pelos algoritmos `insertionSort` e `selectionSort`, nesta ordem.
Cada movimentação efetiva de um número no vetor deve ser contabilizada. Os algoritmos devem ser implementados de maneira a realizar o menor número de trocas possível.

Entrada

A primeira entrada é um número natural n, $1 \leq n \leq 1000$, que representa o tamanho do vetor de entrada. A próxima linha contém os elementos do vetor, sempre fornecidos da primeira posição até a última, e separados por um único espaço em branco entre si.

Saída

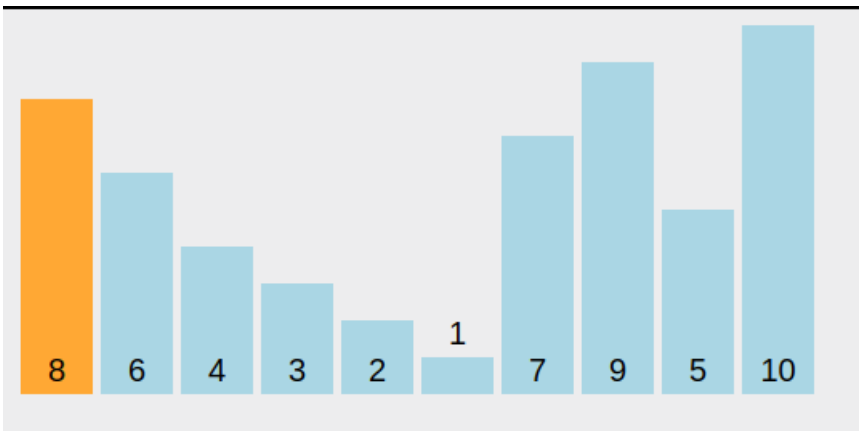
A saída consiste de uma única linha que contém a diferença entre o número de trocas realizadas pelo `insertionSort` e pelo `selectionSort`, nesta ordem.

Exemplos

Entrada	Saída
20 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52	19

Entrada	Saída
20 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33	199

Entrada	Saída
10 8 6 4 3 2 1 7 9 5 10	23



6 Ordenação

 (+)

Escreva um programa $\mathbb{C}$ para, a partir de dois números naturais dados, ℓ e k, com $0 < \ell < k \leq n$, imprima a soma do ℓ-ésimo menor elemento do vetor com o k-ésimo menor elemento do vetor, considerando-se que o referido vetor possui n elementos.

Entrada

A primeira linha da entrada contém o número natural n, que representa o tamanho do vetor, com $1len \leq 1000$.
A segunda linha da entrada contém os n elementos do vetor, sempre apresentados a partir da primeira posição até a última posição, ou seja, das posições $1, 2, 3, \ldots, n$, nesta ordem.
Por fim, a terceira linha da entrada contém os valores de ℓ e k, nesta ordem, e separados por um único espaço em branco.

Saída

Seu programa deve imprimir, numa única linha, o valor a soma do ℓ-ésimo menor elemento do vetor com o k-ésimo menor elemento do vetor.

Exemplos

Entrada	Saída
9 9 8 7 6 5 4 3 2 1 3 4	7

Entrada	Saída
6 1 2 3 6 4 4 5 2	6

Entrada	Saída
6 1 5 9 7 5 3 5 5	14

Entrada	Saída
5 1 7 5 3 9 4 3	12

Observação: O valor de cada elemento do vetor está no intervalo de 1 a 10000, inclusive extremos.

Alice In Numberland

7 Prime Numbers and Arrays

 (++)

Consider a array of natural numbers v, with size expressed by $n \in \mathbb{N}^*$, with $1 \leq n \leq 1000$. The natural numbers are distributed in it totally randomly, that is, there is no apparent order.
Each position of the vector v corresponds to the *order* of a particular prime number, where 2 is the prime number of order 1, 3 is the prime number of order 2 and so on.
A $\mathbb{C}$ computer program must be developed: it will be able to print the prime numbers corresponding to the elements of v, showing them in ascending order.

Input

The first line of the entry will contain the value of n.
The second line will contain the values of the elements of v, always displayed from the first element towards the last element.

Output

The output should display, in just one line, the prime numbers corresponding to the elements of v, in ascending order, and separated by a single blank space between them.

Examples

Entrada	Saída
10 10 2 7 8 5 4 3 1 6 9	2 3 5 7 11 13 17 19 23 29

Entrada	Saída
7 35 140 18 150 28 59 15	47 61 107 149 277 809 863

Entrada	Saída
54 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 1 2 3 4 5 6 7 8 9 10 21 22 23 24 25 26 27 28 29 11 12 13 13 14 16 17 18 19 20	2 3 5 7 11 13 17 19 23 29 31 37 41 41 43 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 193 197 199 211 223 227 229 233 239 241 251

8 Máquina de Café

 (+++)

O Sr. Vanila é dono de uma sofisticada cafeteria na centro da cidade – a StarCoffee – no dia da Proclamação da República, decidiu que irá distribuir cafés grátis como campanha de *marketing* durante todas as 24h deste dia.

Ele decidiu instalar p máquinas de café na StarCoffee. Cada uma das máquinas pode servir uma pessoa por minuto. Ele recebeu a lista de n pessoas que irão visitar sua cafeteria, todos com hora marcada.

O momento da chegada de cada pessoa é denotado por dois números naturais (h, m), onde h é a hora da visita e m é o minuto da visita naquela hora[2].

Devido à tecnologia, no século XXI somos todos(as) *impacientes*, e, assim, se uma pessoa tiver que esperar para obter o seu café após sua chegada na StarCoffee, a pessoa fica **brava**. O Sr. Vanila está preocupado com o custo desta campanha de *marketing* e quer configurar o número mínimo de máquinas de café para que nenhuma pessoa fique brava durante sua estada na StarCoffee. Suponha, para simplificação da resolução, de que não há a menor possibilidade de que qualquer uma das máquinas instaladas apresente defeito durante todo o dia da promoção.

Você, como exímio programador de computadores, recebeu a tarefa de ajudá-lo a encontrar o número mínimo de máquinas de café necessárias para que todos(as) os(as) participantes da campanha de *marketing* fiquem felizes, ou seja, **não bravos(as)**).

Entrada

A primeira linha da entrada contém $t \in \mathbb{N}^*$, o número de casos de testes, com $1 \leq t \leq 100$.

Cada grupo de $(n+1)$ linhas seguintes corresponde a um caso de teste, onde n representa o número de pessoas a serem servidas naquele dia, com $n \in \mathbb{N}^*$ e $n \leq 10.000$.

A primeira linha de cada caso de teste contém o valor de n. As n linhas seguintes contém dois números naturais, h e m, separados por um espaço em branco, representando cada cliente específico.

[2] O horário definido por h:m representa o exato momento em que a pessoa entra na cafeteria e se coloca à frente de uma das máquinas de café disponíveis, onde $0 \leq h \leq 23$ e $0 \leq m \leq 59$.

Saída

Para cada caso de teste fornecido, o programa deve imprimir, numa única linha, o número mínimo de máquinas de café necessário para atender ao problema anteriormente definido.

Restrições

Lembre-se: Todas as restrições definidas devem ser satisfeitas.

Exemplos

Entrada	Saída
1 7 10 20 5 40 10 20 23 11 5 50 10 30 17 12	2

Entrada	Saída
2 8 8 0 9 10 9 11 8 0 8 0 16 40 8 0 16 41 10 13 0 13 1 13 2 13 3 13 0 13 1 13 0 13 0 13 0 13 4	4 5

Entrada	Saída
1 30 9 20 9 20 9 20 9 25 9 22 9 20 9 20 9 20 9 21 9 23 9 20 9 22 9 20 9 21 9 24 9 20 9 21 9 23 9 22 9 20 9 22 9 25 9 22 9 20 9 20 9 23 9 25 9 22 9 24 9 25	12

9 Olimpíadas

(+++)

O Comitê Olímpico Internacional (COI) está visitando as cidades candidatas a sediar as Olimpíadas de 2032 e, desta vez, Goiânia é uma das cidades concorrentes, mas a competição entre as cidades candidatas está muito acirrada, pois Buenos Aires está sendo considerada uma forte candidata.
O COI tem um conjunto de exigências que devem ser obedecidas pelas cidades candidatas, como: (1) boas *arenas* para os jogos (ginásios, campos de futebol, pistas de atletismo, parque aquático,...); (2) bons alojamentos; (3) um plano para o tráfego de veículos durante os jogos, etc.
Durante sua visita à Goiânia, o COI colocou ainda mais uma exigência: a demonstração da qualidade dos sistemas de informática. Especificamente, o COI quer que a organização local demonstre a sua capacidade em informática produzindo um programa de computador que gere a classificação final dos países, considerando o número de medalhas recebidas pelos atletas de cada país participante da futura olimpíada.
Para ser justo, o COI abriu um concurso que permite que equipes de estudantes de quaisquer faculdades, centros universitários e universidades (públicas e privadas) apresentem seus *programas de computador* para cumprir a tarefa. O programa que melhor resultado obtiver numa *bancada de testes* a ser proposta pelo COI, em momento oportuno, será o escolhido.

Tarefa

Você está na equipe que o INF/UFG designou para *vencer* este concurso.
Sua tarefa é escrever um programa $\mathbb{C}$ que, dada a informação dos países que receberam medalhas de ouro, prata e bronze em cada modalidade esportiva, gere a lista de classificação dos países na competição.
Nesta tarefa, os países serão identificados por números inteiros: $1, 2, 3, \ldots, n$, com $2 \leq n \leq 200$.
O melhor colocado deve ser o país que conseguiu o maior número de medalhas de ouro, havendo empate entre dois ou mais países, o melhor colocado é o país que conseguiu o maior número de medalhas de prata. Novamente havendo empate, o melhor colocado é o país que recebeu o maior número de medalhas de bronze. Por fim, se ainda assim houver empate, o melhor classificado é o que tem o maior número de identificação.

Entrada

A primeira linha da entrada contém dois números naturais n e m, separados por um único espaço em branco, e indicando, respectivamente, o número de países e número de modalidades esportivas envolvidas na competição ($1 \leq m \leq 50$). Os países são identificados por numeros inteiros de 1 a n.
Cada uma das m linhas seguintes contém três números inteiros O, P e B, separados por um único espaço em branco entre eles, representando os identificadores dos países cujos atletas receberam, respectivamente, medalhas de ouro, prata e bronze.

Assim, se uma das *m* linhas contém os números 3 2 1, significa que nessa modalidade a medalha de ouro foi ganha pelo país cuja identificação é 3, a de prata pelo país cuja identificação é 2 e a de bronze pelo país cuja identificação é 1.
Uma linha com 10 10 10, significa que o país cuja identificação é 10 recebeu as três medalhas daquela modalidade (ouro, prata e bronze).

Saída

Seu programa deve imprimir, na saída padrão, uma única linha contendo *n* números, separados por um único espaço em branco entre eles, representando os países na ordem decrescente de classificação, da esquerda para a direita.

Exemplos

Entrada	Saída
2 2 2 1 2 1 2 2	2 1

Entrada	Saída
4 3 3 2 1 4 3 1 4 3 1	4 3 2 1

Entrada	Saída
3 3 3 1 2 2 3 1 1 2 3	3 2 1

10 Corrida Escolar

 (+++)

Na escola onde "*Nelsinho*" há, uma vez por ano, uma tradicional corrida de estudantes ao redor de seu prédio principal.
Apesar, evidemente, de haver estudantes dos mais variados períodos, todos(as) os(as) alunos(as) da escola são convidados(as) a participar, sendo impossível que todos participem da mesma corrida, mas em diferentes categorias.
Para contornar esse problema, os(as) professores(as) cronometram o tempo que cada aluno(a) consome para dar cada uma das voltas ao redor do prédio principal, e depois comparam os tempos para descobrir a classificação final, de acordo com a categoria a qual pertence cada estudante.

Tarefa

Sua tarefa é, sabendo o número de competidores(as), o número de voltas de que consistiu a corrida e os tempos de cada aluno(a) competidor(a), descobrir quem foi o(a) aluno(a) vencedor(a), para que ele(a) possa receber uma medalha comemorativa.
Há, obviamente, duas categorias distintas: masculina (M) e feminina (F).

Entrada

A primeira linha da entrada contém dois naturais, n e m, representando o número de competidores(as) e o número de voltas da corrida, respectivamente, sendo que $(1 \leq n \leq 200)$ e $(1 \leq m \leq 5)$.
Na sequência, cada linha representa um competidor(a) e, portanto, haverá n linhas representando os competidores(as).
Em cada linha haverá $(m+1)$ números naturais: o primeiro será 1 para indicar que se trata de um *menino* e 2 para indicar se trata de uma *menina*, e os m seguintes indicam o tempo consumido em cada uma das voltas dadas por aquele(a) competidor(a). O tempo t é expresso em segundos e, por isso, garante-se que

não houve dois(duas) competidores(as) que gastaram o mesmo tempo para completar a corrida inteira, ou seja, o tempo total de todas as suas voltas.
Para simplificar a solução do problema considera-se que nenhum(a) dos(as) competidores(as) desistiu da corrida, ou seja, todos(as) concluiram as m voltas da corrida da qual participou. Sabe-se também que $1 \leq t \leq 1000$.

Saída

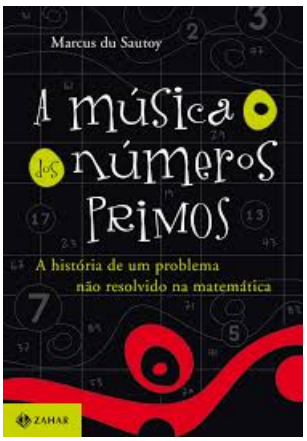
A saída deve consistir de dois números naturais, separados por um único espaço em branco entre eles, que correspondem, respectivamente, ao vencedor da prova masculina à vencedora da prova feminina.

Exemplos

Entrada	Saída
2 3 1 2 1 2 2 1 2 3	1 2

Entrada	Saída
4 3 1 3 2 1 1 4 3 1 2 1 4 2 2 6 3 1 7	1 3

Entrada	Saída
3 3 2 3 5 6 2 1 2 3 1 1 1 1	3 2



11 Fermat e os Números Primos

 (+++)

Pierre de Fermat (1607 – 1665) foi advogado e magistrado francês que tinha, como *hobby*, estudar Matemática. Sua biografia mostra que ele era um homem cortês e amigável, mas que não era proponso a relacionamentos próximos: preferia comunicar-se por meio de cartas.
Numa carta datada de 25 de dezembro de 1640, ele escreveu para seu amigo, o padre Martin Mersenne (1588 – 1648), que havia feito uma descoberta "*maravilhosa*": de que alguns números primos podiam ser escritos como a soma do quadrado de dois números naturais.
Por exemplo:

- $2 = 1 + 1 = 1^2 + 1^2$;
- $5 = 1 + 4 = 1^2 + 2^2$;
- $13 = 4 + 9 = 2^2 + 3^2$;
- $17 = 1 + 16 = 1^2 + 4^2$.

Por outro lado, identificou que outros números primos, como 3, 7, 11, 19 e 23 não podiam ser decompostos desta maneira.
Assim, os números primos que podem ser decompostos como a soma do quadrado de dois números são chamados, atualmente, de "*Números Primos de Fermat*" ou, simplesmente, "*Primos de Fermat*".

Tarefa

Sua tarefa é escrever um programa $\mathbb{C}$ que seja capaz de imprimir os "*Primos de Fermat*" que são menores ou iguais a um certo número n, $n \in \mathbb{N}^*$, fornecido.

Entrada

A primeira linha da entrada contém o número de casos de teste t, sendo que $1 \leq t \leq 50$.
Na linha seguinte haverá os t números, sempre separados por um único espaço em branco entre eles. Sabe-se também que $1 \leq t_i \leq 10^6$, com $1 \leq i \leq t$.

Saída

Cada um dos *t* casos de teste deverá ter sua saída impressa numa única linha, como os "*Primos de Fermat*" separados entre si por um único espaço em branco.
Nos casos de teste de ordem ímpar (1º, 3º, 5º, 7º, ...), os números devem ser impressos em ordem *crescente*, enquanto que nos casos de teste de ordem par (2º, 4º, 6º, 8º, ...) eles devem ser impressos em ordem *decrescente* – veja os exemplos a seguir.

Exemplos

Entrada	Saída
1 50	2 5 13 17 29 37 41

Entrada	Saída
3 50 80 100	2 5 13 17 29 37 41 73 61 53 41 37 29 17 13 5 2 2 5 13 17 29 37 41 53 61 73 89 97

Entrada	Saída
7 28 23 30 12 25 13 11	2 5 13 17 17 13 5 2 2 5 13 17 29 5 2 2 5 13 17 13 5 2 2 5

Observação:
No primeiro exemplo, como há apenas um caso de teste, os números, na saída, devem ser impressos em ordem crescente.
No segundo exemplo há três casos de teste: o primeiro (50) e o terceiro (100) devem ter seus números impressos, na saída, em ordem crescente, enquanto que o segundo (80) deve ter seus números impressos em ordem decrescente.