Universidade Federal de Viçosa
***Campus* Rio Paranaíba**
Sistemas de Informação
Prof. João Batista Ribeiro

SIN343 – Desafios de Programação
Prova 2 – Maratona 2
24/01/2020

Observações:
- Uma máquina por equipe e não é permitido pegar material emprestado durante a maratona.
- Dispositivos eletrônicos não podem ser utilizados e devem ser guardados.
- O acesso à Internet é proibido até o final da maratona, exceto ao IP do BOCA.
- Este caderno de problemas contém 6 problemas e as páginas estão numeradas de 1 a 6.
- IP do Servidor: http://200.17.76.19/boca/

Equipe:			
Matrícula 1:		Nome 1:	
Matrícula 2:		Nome 2:	
Matrícula 3:		Nome 3:	

Problema a
Subsequência

Arquivo-fonte: sub.(c | cpp | java)

Uma sequência de N inteiros positivos (0 < N < 100 000), cada um deles menor ou igual a 10000, e um inteiro positivo S (S < 100 000 000) são dados. Escreva um programa para encontrar o tamanho mínimo da subsequência de elementos consecutivos dessa sequência, cuja soma seja maior ou igual a S.

Entrada

A entrada começa com um inteiro C indicando o número de casos de teste. Para cada um deles seu programa deve ler os números N e S, na primeira linha. Na segunda linha serão dados os números da sequência.

Saída

Para cada caso seu programa deve imprimir, por linha, a mensagem "Caso #:", onde # indica o número do caso iniciando em 1, um espaço em branco e em seguida imprima o resultado da menor subsequência encontrada, se não houver nenhuma tal subsequência, imprima 0. Toda linha termina com um salto de linha.

Exemplo de Entrada	**Exemplo de Saída**
2 10 15 5 1 3 5 10 7 4 9 2 8 5 11 1 2 3 4 5	Caso 1: 2 Caso 2: 3

Problema b
Limite de Velocidade
Arquivo-fonte: limite.(c | cpp | java)

Bill e Ted estão viajando de carro. Mas o hodômetro do carro não está funcionando, então eles nem sabem quantas milhas já viajaram. Felizmente, Bill tem um cronômetro, então eles podem anotar a velocidade e o tempo total de viagem. Infelizmente, sua estratégia de anotação é meio estranha, então eles agora precisam de ajuda para calcular a distância total percorrida. Você deve escrever um programa para realizar esse cálculo. Por exemplo, se as anotações mostram:

Velocidade em milhas/h	Tempo passado em horas
20	2
30	6
10	7

isso significa que eles dirigiram 2 horas a 20 milhas/h, depois 6-2=4 horas a 30 milhas/h, e então 7-6=1 hora a 10 milhas/h. A distância total percorrida é então (2)(20) + (4)(30) + (1)(10) = 40 + 120 + 10 = 170 milhas. Note que o tempo passado é contado sempre desde o início da viagem, e não desde a última anotação.

Entrada
A entrada consiste de um ou mais conjuntos de teste. Cada conjunto começa com uma linha contendo um inteiro n, $1 \leq n \leq 10$, seguido por n pares de valores, um por linha. O primeiro valor do par, v, é a velocidade em milhas/h e o segundo valor, t, é o tempo total já passado. Ambos v e t são inteiros, $1 \leq v \leq 90$ e $1 \leq t \leq 12$. Os valores de t estão sempre em ordem estritamente crescente. Um valor -1 para n indica o fim da entrada.

Saída
Para cada conjunto da entrada, imprima a distância percorrida, seguida de um espaço, e depois a palavra “miles”. Lembre-se do salto de linha na última linha.

Exemplo de Entrada	Exemplo de Saída
3 20 2 30 6 10 7 2 60 1 30 5 4 15 1 25 2 30 3 10 5 -1	170 miles 180 miles 90 miles

Problema c
O problema 3n+1
Arquivo-fonte: ciclo.(c | cpp | java)

Considere o seguinte algoritmo para gerar uma sequência de números. Comece com um inteiro N. Se N é par, divida-o por 2. Se N é ímpar, multiplique-o por 3 e some 1. Repita esse processo com o novo valor de N, terminando quando N=1. Por exemplo, a seguinte sequência de números será gerada para N=22:

22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

É conjecturado (mas ainda não provado) que este algoritmo terminará em N=1 para cada inteiro N. Ainda assim, a conjectura é válida para todos os números inteiros até pelo menos 1000000. Para uma entrada N, o comprimento do ciclo de N é o número de números gerados a partir do valor inicial incluindo-o. No exemplo acima, o comprimento do ciclo de 22 é 16. Dados quaisquer dois números x e y, você deverá determinar o ciclo de maior comprimento dentre os números do intervalo x e y, incluindo os valores x e y.

Entrada
A entrada consiste de uma série de pares de inteiros x e y, um par de inteiros por linha. Todos os inteiros serão menores do que 1000000 e maiores do que 0.

Saída
Para cada par de inteiros x e y, a saída será formada por x e y na mesma ordem que eles aparecem na entrada e pelo valor do ciclo de maior comprimento deste intervalo. Estes três valores deverão estar na mesma linha e separados por um espaço em branco. Cada linha de entrada terá uma linha de saída. Lembre-se do salto de linha na última linha.

Exemplo de Entrada	**Exemplo de Saída**
1 10 100 200 201 210 900 1000	1 10 20 100 200 125 201 210 89 900 1000 174

Problema d
Campo Minado
Arquivo-fonte: minas.(c | cpp | java)

Você já jogou campo minado? Esse clássico joguinho vem com um sistema operacional que eu não recordo o nome agora. O objetivo do jogo é encontrar onde todas as minas estão localizadas dentro de um campo N x M. O jogo mostra um *número dica* em um quadrado que diz quantas minas são adjacentes àquele quadrado. Cada quadrado tem no máximo oito quadrados adjacentes. O campo 4 x 4 à esquerda contém duas minas, cada uma representada pelo carácter '*****'. Se representarmos o mesmo campo pelos *números dica* descritos acima, nós teríamos um campo como o representado à direita.

* . . .	* 1 0 0
. . . .	2 2 1 0
. * . .	1 * 1 0
. . . .	1 1 1 0

Entrada
A entrada consiste de um número aleatório de campos. A primeira linha de cada campo contém dois inteiros N e M ($0 < N, M \leq 100$) que representam o número de linhas e colunas do campo, respectivamente. Cada uma das próximas N linhas contém exatamente M caracteres, representando o campo. Quadrados seguros serão denotados por um '**.**' e quadrados com minas por um '*****', ambos sem aspas simples. A primeira linha onde N = M = 0 representará o fim da entrada e não deverá ser processada.

Saída
Para cada campo, imprima a mensagem "Campo #X:" em uma linha, onde X representa o número do campo começando do 1. As próximas N linhas deverão conter o campo com os caracteres '**.**' substituídos pelos números de minas adjacentes àquele quadrado. Deve haver uma linha em branco entre as saídas de cada campo.

Exemplo de Entrada	**Exemplo de Saída**
4 4 *… …. .*.. …. 3 5 **… ….. .*... 0 0	Campo #1: *100 2210 1*10 1110 Campo #2: **100 33200 1*100

Problema e
BOCA
Arquivo-fonte: boca.(c | cpp | java)

Julgar campeonatos de programação é uma árdua tarefa e sistemas como o BOCA que automatizam o processo de julgamento são uma grande ajuda. Você é parte de um grupo que tenta desenvolver um *software* de gerenciamento de campeonatos baseado em técnicas de projeto modular. O componente que você trabalha é aquele que calcula os *scores* dos times participantes e determina o vencedor. Você receberá os resultados de vários times, e deve determinar o campeão.

Há dois componentes no *score* de um time. O primeiro é o número de problemas resolvidos. O segundo é a penalidade, que reflete o tempo e as submissões incorretas antes de se resolver o problema. Para cada problema resolvido corretamente, a penalidade é igual ao tempo em que o problema foi resolvido, mais 20 minutos por submissão incorreta. Nenhuma penalidade é dada para problemas que não foram resolvidos.

Assim, se um time resolve o problema 1 na sua segunda submissão, no tempo 20 minutos, o time recebe 40 minutos de penalidade. Se ele submete o problema 2 três vezes, mas não consegue resolvê-lo, nenhuma penalidade é acrescentada. Se ele submete o problema 3 uma única vez e o resolve no tempo 120 minutos, então ele recebe 120 minutos de penalidade. Seu *score* total será 2 problemas resolvidos com 160 pontos de penalidade. O vencedor é o time que resolver a maior quantidade de problemas. Se houver empate no maior número de problemas resolvidos, então o vencedor entre eles é o time com a menor penalidade total.

Entrada
Para o campeonato que seu programa deve julgar, há quatro problemas. Garantidamente não haverá empate depois de contadas as penalidades dos times. A primeira linha da entrada contem um inteiro N representando o número de times. As próximas N linhas contém cada uma, o nome do time, que não contém espaço em branco. Em seguida, para cada um dos quatro problemas, o número de submissões feitas pelo time e o tempo no qual conseguiu uma submissão correta (ambos inteiros). Se o time não conseguiu resolver o problema, o tempo será zero. O número de submissões será pelo menos um, se o problema foi resolvido pelo time.

Saída
A saída consiste em uma única linha, contendo o nome do time vencedor, o número de problemas resolvidos por esse time, e sua penalidade total.

Exemplo de Entrada	**Exemplo de Saída**
4 Stars 2 20 5 0 4 190 3 220 Rockets 5 180 1 0 2 0 3 100 Penguins 1 15 3 120 1 300 4 0 Marsupials 9 0 3 100 2 220 3 80	Penguins 3 475

Problema f
Fila de Impressão
Arquivo-fonte: fila.(c | cpp | java)

A única impressora do curso de Sistemas de Informação está trabalhando pesado ultimamente. Em certos momento há centenas de documentos na fila de impressão, e você tem que esperar horas para conseguir imprimir até mesmo uma simples página. Já que alguns documentos são mais importantes que outros, o coordenador inventou e implementou um sistema simples de prioridade para controlar a fila de impressão. Agora, a cada documento é associado uma prioridade entre 1 e 9 (sendo 9 a mais alta prioridade e 1 a mais baixa), e a fila de impressão funciona da seguinte forma:

- O primeiro documento J da fila de impressão é retirado da fila;
- Se houver algum documento na fila de impressão com prioridade mais alta que a do documento J, então J é colocado no final da fila de impressão sem ser impresso;
- Caso contrário, o documento J é impresso (e não é colocado de volta na fila de impressão)

Desta forma, todas os documentos importantes da coordenação são impressos rapidamente. E outros documentos menos importantes podem ser forçados a esperar um bom tempo até serem impressos, mas fazer o que, é a vida… Você quer saber quando seu documento será efetivamente impresso. Então você precisa fazer um programa para determinar isso. Serão informadas ao programa a fila corrente (e uma lista de prioridades) e a posição de seu documento na fila. Nenhum documento novo vai chegar. O programa deve então calcular quanto tempo demora para seu documento ser impresso, assumindo que nenhum documento adicional será colocado na fila. Para simplificar, considere que todo documento leva exatamente 1 minuto para ser impresso, e que retirar e colocar documentos na fila são instantâneos.

Entrada
A entrada começa com um número inteiro que indica o número de casos de teste (até 100). Cada caso de teste consiste em:

- uma linha com dois inteiros N e M, sendo N o número de trabalhos na fila ($1 \leq N \leq 100$) e M a posição de seu trabalho ($0 \leq M \leq N-1$).
- em seguida, uma linha com N inteiros na faixa 1 a 9, que são as respectivas prioridades dos documentos na fila.

Saída
Para cada caso de teste, escreva uma linha com um inteiro, o número de minutos que se passam até que seu documento seja completamente impresso.

Exemplo de Entrada	**Exemplo de Saída**
3 1 0 5 4 2 1 2 3 4 6 0 1 1 9 1 1 1	1 2 5