UNIVERSIDADE FEDERAL DE ITAJUBÁ

ECOX21 - MARATONA DE PROGRAMAÇÃO I

Profs. Edmilson Marmo e Luiz Olmes

Warmup 3

23/10/2024

Regras

1. Há 6 problemas que devem ser resolvidos no tempo estipulado.
2. Os dados de entrada devem ser lidos a partir da entrada padrão.
3. Os dados de saída devem ser escritos na saída padrão.
4. Quando uma linha contém vários valores, eles estarão separados por um único espaço. Não há espaços extras na entrada/saída. Não há linhas em branco na entrada.
5. A entrada e a saída usam o alfabeto latino. Não haverá letras com til, acentos, tremas ou outros sinais diacríticos.
6. Todas as linhas da entrada e da saída, incluindo a última, contêm o caractere de fim de linha.
7. O código fonte de cada solução deve ser enviado pelo Boca: `boca.unifei.edu.br`

Ambiente de Testes

A correção das soluções enviadas é realizada no sistema operacional Red Hat Enterprise Linux, versão 8.6 (Ootpa), usando os seguintes compiladores/interpretadores:

```
C: gcc versão 8.5.0 20210514 (Red Hat 8.5.0-10)
C++: g++ versão 8.5.0 20210514 (Red Hat 8.5.0-10)
Java: openjdk versão 1.8.0_342
Python: python3 versão 3.6.8
```

Limites

```
Memória (C, C++, Python): 1GB
Memória (Java): 1GB + 100MB stack
Tamanho máximo do código fonte: 100KB
Tamanho máximo do arquivo executável: 1MB
```

Códigos que extrapolem os limites permitidos receberão *Runtime Error* como resposta.

Comandos de Compilação

```
C: gcc -g -O2 -std=gnu11 -static -lm
C++: g++ -g -O2 -std=gnu++17 -static -lm
Java: javac
```

Códigos C/C++

- O programa deve retornar zero, executando, como último comando, `return 0` ou `exit(0)`.
- Para entradas grandes, objetos `iostream` podem ser lentos, devido a questões de sincronização de buffer com a biblioteca `stdio`. Recomenda-se desabilitar este mecanismo de sincronização em programas que empreagam `std::cin` e `std::cout` através dos seguintes comandos:

  ```
  std::ios::sync_with_stdio(0);
  std::cin.tie(0);
  ```

  Note que, neste caso, deve-se evitar usar `printf` e `scanf` no mesmo programa, pois a separação dos buffers pode levar a resultados inesperados.

Códigos Java

- O programa não deve estar encapsulado em um `package`.
- Para cada problema, o arquivo `.java` e a `public class` devem ter o mesmo nome `basename` mostrado no Boca.
- Comando de execução: `java -Xms1024m -Xmx1024m -Xss100m`

Códigos Python

- Apenas Python 3 é suportado. Python 3 não é compatível com Python 2.
- **Atenção**: não é garantido que soluções escritas em Python executarão dentro do tempo limite especificado para cada problema.
- Comando de execução: `python3`

Problema $\mathcal{A}$

JOLLY JUMPERS

Timelimit: 1s

Uma sequência de $n > 0$ inteiros é chamado de *jolly jumpers* se os valores absolutos das diferenças entre os elementos sucessivos abrangem todos os valores possíveis entre 1 e $n - 1$. Por exemplo,

1 4 2 3

é uma sequência *jolly jumpers*, porque as diferenças absolutas são 3, 2 e 1, respectivamente. Esta definição implica que qualquer sequência de um único inteiro é um *jolly jumpers*. Escreva um programa para determinar se cada sequência da entrada é *jolly jumper*.

Entrada

A entrada contém vários casos de teste. Cada caso de teste possui duas linhas. A primeira linha contém um inteiro N ($0 \leq N \leq 3000$) que indica o número de elementos da sequência. A próxima linha contém os N números da sequência, cada número separado por um espaço em branco. O final da entrada é indicado por $N = 0$.

Saída

Para cada caso de teste seu programa deve produzir uma única linha de saída dizendo se a sequência é *jolly jumpers* (`SIM`) ou não é *jolly jumpers* (`NAO`).

Exemplos

Entrada	*Saída*
5 5 1 4 2 3 6 5 1 4 2 -1 6 0	SIM NAO

Problema $\mathcal{B}$

OPERADORES LÓGICOS

Timelimit: 1s

Lucas e Mateus só querem saber de jogar video-game. Seu pai resolveu dar um castigo para os dois: aprender o funcionamento dos operadores lógicos. Este castigo tem uma razão. Se continuarem só querendo jogar, terão que aprender a programar. Então, nada melhor que começarem com os operadores lógicos, utilizados em todas as linguagens de programação.

Toda vez que Lucas e Mateus quiserem jogar, eles deverão resolver uma série de problemas lógicos. Somente após, caso tenham acertado, poderão brincar.

O pai dos garotos pediu a sua ajuda, já que não quer perder tempo resolvendo os problemas. Assim, a partir das operações, ele quer um programa que gere o gabarito dos resultados. Detalhe importante, os garotos já sabem trabalhar na base binária. Portanto, as operações serão realizadas na base 2.

Os operadores lógicos que os meninos estão aprendendo são: NOT (operador unário de negação), AND (operador binário de conjunção), OR (operador binário de disjunção inclusiva), XOR (operador binário de disjunção exclusiva), NAND (operador binário de negação da conjunção), NOR (operador binário de negação da disjunção inclusiva) e XNOR (operador binário de negação da disjunção exclusiva).

Entrada

A entrada é composta por vários conjuntos de teste. A primeira linha contém um número B (detalhe: em binário) que corresponde a quantidade de operações que serão analisadas. As demais B linhas contém uma operação binária aplicada sobre um (no caso do operador unário) ou dois números (no caso dos operadores binários) binários. Os números binários têm, exatamente, 4 dígitos binários, mesmo que sejam usados zeros a esquerda do número. Os operadores são representados por letras maiúsculas.

Saída

Para cada caso de teste haverá apenas uma linha contendo um único número binário, representando o resultado da operação. O número deve conter, exatamente, quatro dígitos.

Exemplos

Entrada	*Saída*
11 0011 AND 1100 0011 OR 1100 0011 XOR 1100	0000 1111 1111

Problema C

NÃO ACREDITO!

Autoria: Edmilson Marmo
Timelimit: 1s

Existe um jogo matemático que pode ser facilmente implementado em um computador. Inicia-se sempre com um inteiro positivo S. Caso ele tenha mais de um dígito, computa-se o produto dos seus dígitos, repetindo este processo até que não seja mais possível, o que significa que um único dígito foi alcançado. Por exemplo, se começar com 95, computa-se $9 \times 5 = 45$. Como 45 também possui mais de um dígito, repete-se o procedimento com $4 \times 5 = 20$. Continuando com 20, calcula-se $2 \times 0 = 0$. Chegando a zero, que é um número de apenas um dígito, o jogo termina.

Tenho certeza que você não está acreditando que este problema foi cobrado neste *warmup*. Então, não perca tempo e implemente o jogo!

Entrada

Cada linha conterá um único inteiro S ($1 \leq S \leq 100000$), designando o valor inicial. O valor de S não terá nenhum 0 não-significativo. O valor $S = 0$ indica o fim da entrada.

Saída

Para cada valor de entrada diferente de zero, você deve imprimir uma linha na saída que expresse a sequência ordenada de valores que são obtidos durante os cálculos do jogo, começando com o valor original. *Não deve haver espaço após o último elemento impresso na linha.*

Exemplos

Entrada	*Saída*
95 162 4 0	95 45 20 0 162 12 2 4

Problema $\mathcal{D}$

CLASSES DE EQUIVALÊNCIA

Autoria: Edmilson Marmo
Timelimit: 1s

Uma relação R em um conjunto S é denominada relação de equivalência se ela satisfaz todas as três propriedades a seguir:

1. **Reflexividade**. Para todo $a \in S, a\ R\ a$.
2. **Simetria**. Para todos $a, b \in S, a\ R\ b \Leftrightarrow b\ R\ a$.
3. **Transitividade**. Para todos $a, b, c \in S$, se $a\ R\ b \wedge b\ R\ c$, então $a\ R\ c$.

Um importante resultado é que cada relação de equivalência induz um particionamento do conjunto (em que a relação é definida) em subconjuntos disjuntos e não vazios denominados classes de equivalência.

Pede-se desenvolver um programa que verifique se uma determinada relação é uma relação de equivalência e, neste caso, apresentar a quantidade de classes de equivalência.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém um valor N ($0 \leq N \leq 100$) indicando os elementos do conjunto S (todos os números de 0 até $N-1$). A próxima linha apresenta um número M ($1 \leq M < 10000$) indicando a quantidade de pares ordenados pertencentes a R. Cada uma das M linhas seguintes apresenta dois números indicando o respectivo par ordenado. O final da entrada é indicado por $N = 0$.

Saída

Para cada caso de teste da entrada será impresso uma única linha com o número de classes de equivalência, sendo 0 para a relação que não corresponde a uma relação de equivalência.

Exemplos

Entrada	*Saída*
1 1 0 0 5 5 0 0 1 1 2 2 3 3 4 4 3 2 1 2 2 1 0	1 5 0

Problema $\mathcal{E}$

O CRIVO DE ERATÓSTENES

Autoria: Edmilson Marmo
Timelimit: 2s

O Crivo de Eratóstenes é um algoritmo e um método simples para encontrar números primos até um certo valor limite. Segundo a tradição, foi criado pelo matemático grego Eratóstenes, o terceiro bibliotecário-chefe da Biblioteca de Alexandria desde 247. O algoritmo funciona eliminando os múltiplos de cada número primo, deixando apenas os números primos "marcados".

O algoritmo funciona da seguinte maneira:

1. Inicialmente, considere que todos os números de 2 até N são primos.
2. A partir do menor número primo, marque todos os seus múltiplos como não primos.
3. Repita o processo para o próximo número primo não marcado até que todos os números tenham sido processados.
4. Ao final, todos os números que não foram marcados como múltiplos são primos.

Confira um exemplo na tabela a seguir que marca todos os valores de 1 até 100.

Neste problema, você deverá utilizar uma variação do Crivo de Eratóstenes para contar quantos números foram marcados dentro de um intervalo $[A, B]$.

Entrada

Cada linha conterá dois números inteiros positivos A e B ($2 \leq A \leq B \leq 10^6$), designando um intervalo. Os valores $A = B = 0$ indica o fim da entrada.

Saída

Para cada intervalo, o programa deverá imprimir a quantidade de elementos marcados pelo algoritmo.

Exemplos

Entrada	*Saída*
2 10 2 100 0 0	5 74

Problema $\mathcal{F}$

FUGINDO DO CHEFE

Timelimit: 2s

Ao acordar hoje de manhã, ainda com muito sono, você percebeu que a sua cama é um ambiente muito mais aconchegante do que o escritório onde trabalha. Assim sendo, você decidiu que gostaria de ficar em casa para maratonar a sua série favorita. Você resolveu ligar para o seu chefe, com uma voz fraca e desanimada, dizendo que estava doente. Seu chefe é uma pessoa bem rígida, porém muito comprensivo com essas situações. Por telefone mesmo, ele te desejou melhoras e disse que você poderia ficar em casa hoje. Com o passar do dia, você descobriu que não tem nada para comer em casa e precisa sair para fazer compras. Porém você tem muito medo de que seu chefe o veja na rua, o que seria demissão. Então você precisa saber se existe alguma opção de rota segura para chegar ao mercado.

Você sabe que seu chefe é muito metódico e fica apenas em dois lugares: a residência dele ou o escritório. Ao ir de um lugar para outro, ele sempre usa o menor caminho possível. Assim, você não deve passar pelos mesmos locais que ele, pois ele não pode te ver fora de casa. Dessa forma, sua missão é conseguir chegar da sua casa até o mercado, sem cruzar com o seu chefe em nenhum momento.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém seis inteiros P, T, C, E, S e M, onde:

- P: número de pontos (locais) da cidade ($1 \leq P \leq 100$).
- T: número de conexões existentes entre os pontos ($0 \leq T \leq 5000$).
- C: o ponto onde seu chefe mora ($1 \leq C \leq P$).
- E: o ponto onde se situa o escritório ($1 \leq E \leq P$).
- S: o ponto onde você mora ($1 \leq S \leq P$).
- M: o ponto onde se situa o mercado ($1 \leq M \leq P$).

Seguem-se T linhas contendo a descrição da cidade. Cada uma dessas linhas contém três inteiros A, B e D, denotando que a via que liga os pontos A e B possui distância D ($0 \leq A, B, \leq P; 1 \leq D \leq 200$). Você pode assumir que todas as vias são de mão dupla, e não existe mais de uma via conectando diretamente dois pontos quaisquer. A entrada termina com fim de arquivo (`EOF`).

Saída

Para cada caso de teste, seu programa deve imprimir uma linha contendo um inteiro que representa o custo do menor caminho para ir de sua casa até o mercado, evitando o seu chefe. Se não for possível percorrer este trajeto, imprima -1 como resposta.

Exemplos

Entrada	*Saída*
3 2 2 3 1 3 1 2 4 2 3 4 3 2 2 3 3 3 1 2 4 2 3 4 4 3 2 3 1 4 1 2 4 2 3 4 1 4 10	-1 -1 10