APS – Atividade Prática Supervisionada
Técnicas de Programação

Linguagem C

Prof. Igor Oliveira Borges
igor.borges@anhembi.br

Prof. Paulo Rogerio Nietto
prnietto@anhembi.br

Descrição e Orientação Geral

A APS da disciplina consiste na resolução de uma lista de desafios de lógica de programação e a sua implementação em C.

A atividade pode ser realizada em trios, cabendo ao aluno que enviar a resolução, informar o nome e o RA dos membros integrantes do trio. Isso deve ser feito no campo de comentários do envio da atividade.

A entrega deve ser feita exclusivamente pela unidade web, na atividade destinada a esse fim. Demais envios serão descartados. O trio deverá enviar 3 soluções. Cada solução vale 2,5 pontos. No dia da entrega, cada grupo irá apresentar para a sala a solução de um desafio, sendo que o professor irá escolher o que cada grupo irá apresentar.

A entrega será o envio de um arquivo único compactado em formato ZIP ou RAR com extensão .zip ou .rar. NÃO utilize .7z, etc. Arquivos em formato distinto ao .zip ou ao .rar serão descartados.

Os arquivos entregues na compactação devem ser o código-fonte, ou seja, de extensão .c. Cada desafio solucionado deve corresponder à um arquivo fonte .c. O nome do arquivo deve corresponder a numeração do desafio. Por exemplo, se o trio solucionar o desafio 3 – popularidade, então o arquivo correspondente será: *desafio3.c.* Siga essa nomenclatura para facilitar a correção.

No topo de cada arquivo fonte .c, o comentário com o nome do desafio solucionado, nome e o RA dos membros do trio conforme o formato abaixo:

```
/*
* Desafio 3 - Popularidade
* Fulano de Tal 1   RA: 90909090
* Fulano de Tal 2   RA: 12312312
* Fulano de Tal 3   RA: 45645645
*/
```

As partes mais relevantes do algoritmo DEVEM ser comentadas a título de esclarecer a lógica empregada. Por exemplo, no uso de um laço de verificação ou comparação para identificar a menor pontuação. Comente com bom senso o seu código.
Caso um desafio seja resolvido corretamente somente por um trio, este trio terá um (1,0) ponto adicional na nota da APS.

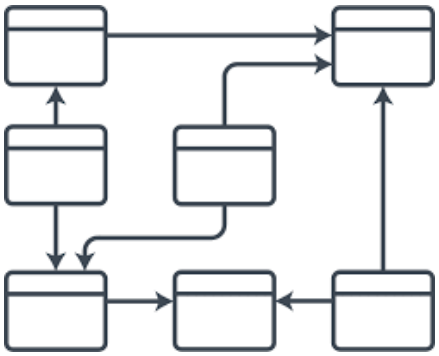
Caso você tenha lido até este item do enunciado, insira um comentário ao final do arquivo do último desafio entregue pela sua dupla com o texto: “li e concordo com os termos da aps”. As 5 primeiras submissões com esta ocorrência receberão um acréscimo de 0.5 pto por este *easter egg*.

Por último, respeite os prazos, siga cada orientação explicitada nesse texto da APS. Cada descrição será utilizada direta ou indiretamente como critério (rubrica) de avaliação. Sendo assim, para que a dupla tire 10, ela terá que demonstrar habilidade tanto na parte técnica, ou seja, capacidade de programar e resolver problemas, quanto atenção, leitura, interpretação de textos e retidão ao cumprimento das regras.

Ao final das aulas de TP, serão destinados os últimos 20 min à discussões sobre estes problemas. Aproveite esse espaço.

Desafio 1 – Vamos implementar um CRUD (*Create, Read, Update e Delete*)?

Neste problema, pede-se para armazenar, gerenciar e buscar por indivíduos definidos por um identificador único (inteiro) e pelas seguintes informações: Primeiro e último nome, data de nascimento e telefone.



Entrada
A entrada será feita por 4 comandos: *add*, *del*, *info* e *query*. A execução é encerrada com a sequência de caracteres: *"000"*.

O comando *''add''* recebe e armazena todos dados do individuo, e retorna erro se já existir individuo com mesmo identificador.

```
add <id> <first_name> <last_name> <birtday> <phone_number>
```

O comando *''del''* remove todos dados relacionados a um determinado identificador, e retorna erro se não existir individuo com o identificador fornecido.

```
del <id>
```

O comando *''info''* imprime todos dados de um determinado identificador, e retorna erro se não existir individuo com o identificador fornecido.

```
info <id>
```

O comando *''query''* realiza uma busca nos indivíduos cadastrados. Conforme as seguintes tags de busca:

```
fn: Primeiro nome
ln: Ultimo nome
bd: Data de nascimento
pn: Telefone


query (<tag>:<valor>)+
```

Saída

O comando *''add''* somente imprime na saída quando ocorre erro na inserção de um individuo, ocorrida na inserção de individuo com identificador duplicado.

O comando *''del''* somente imprime na saída quando o identificador solicitado não existe.

O comando *''info''* imprime todos dados de um determinado identificador, ou imprime erro se não existir individuo com o identificador fornecido.

O comando *''query''* retorna os identificadores que respeitem os critérios da busca na ordem crescente separados por espaços. Em caso de não existir nenhum individuo que respeite a busca, uma linha vazia deve ser impressa.

Exemplo

Entrada:

```
add 123 Roberto Nascimento 01/01/1960 +55-21-0190-0190
add 123 Joao Souza 11/10/2000 103-99
add 09 Andre Matias 01/01/1970 +55-21-0190-0190
add 222 Diogo Fraga 01/06/1967 +55-21-0190-0190
add 99 Seu Barbosa 01/01/1960 +55-21-0190-0190
add 100 Seu Beirada 01/01/1960 +55-21-9999-9999
add 155 Andre Fortunato 02/01/1962 +55-21-0190-0190
query bd:01/01/1960
query bd:01/01/1960 fn:Seu
query bd:01/01/1960 fn:Seu pn:+55-21-9999-9999
info 100
del 99
query fn:XXX
query bd:01/01/1960 fn:Seu
info 99
del 99
000
```

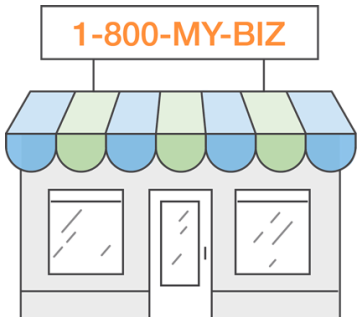
Saída:

```
ID 123 ja cadastrado.
99 100 123
99 100
100
Seu Beirada 01/01/1960 +55-21-9999-9999

100
ID 99 nao existente.
ID 99 nao existente.
```

Desafio 2 – Encontre o telefone

Em alguns lugares é comum lembrar um número do telefone associando seus dígitos a letras. Dessa maneira a expressão MY LOVE significa 69 5683. Claro que existem alguns problemas, uma vez que alguns números de telefone não formam uma palavra ou uma frase e os dígitos 1 e 0 não estão associados a nenhuma letra.



Sua tarefa é ler uma expressão e encontrar o número de telefone correspondente baseado na tabela abaixo. Uma expressão é composta por letras maiúsculas (A-Z), hifens (-) e os números 1 e 0.

Letras	**Número**
ABC	2
DEF	3
GHI	4
JKL	5
MNO	6
PQRS	7
TUV	8
WXYZ	9

Entrada
A entrada consiste de um conjunto de expressões. Cada expressão está sozinha em uma linha e possui C caracteres, onde 1 ≤ C ≤ 30. A entrada é encerrada com *'00000'*.

Saída
Para cada expressão você deve imprimir o número de telefone correspondente.

Exemplo:

Entrada:
1-HOME-SWEET-HOME
MY-MISERABLE-JOB

Saída:
1-4663-79338-4663
69-647372253-562

Desafio 3 – Popularidade

Uma escola muito tradicional está promovendo uma eleição de popularidade, para determinar, naturalmente, quem é o aluno mais popular. Foi definido, então, que cada aluno deverá votar nos alunos de quem gosta. A quantidade de votos dados por cada aluno é variável, isto é, cada aluno pode votar em quantos alunos desejar, de acordo com suas preferências. O vencedor será aquele que receber mais votos, ou seja, aquele para o qual mais alunos indicaram que gostam dele.

Para realizar a eleição, cada aluno receberá uma cédula eleitoral contendo os nomes de todos os alunos da escola (inclusive ele próprio), na qual deverá preencher os quadrados ao lado dos nomes dos alunos que gosta, utilizando caneta esferográfica azul ou preta. Após o término do período de votação, as cédulas serão colocadas numa máquina, a qual é capaz de informar quais quadrados foram preenchidos em cada cédula.

Você, estagiário da escola em questão, ficou responsável por apresentar um protótipo do sistema que recebe as informações da máquina e contabiliza os dados da eleição. Por se tratar de um protótipo, sua tarefa é apenas escrever um programa que, dadas as informações sobre simulações de preenchimento das cédulas, informe quantos votos recebeu o vencedor da eleição.

Deve-se assumir que os alunos da escola são participativos, que todos compareceram à votação e cada um preencheu exatamente uma cédula eleitoral. Pode-se assumir, ainda, que os alunos não sofrem por conflitos internos, de modo que cada aluno gosta de si mesmo e vota em si mesmo. Note, porém, que a relação ``gostar de'' não é simétrica, ou seja, se o aluno A gosta do aluno B, não necessariamente o aluno B gosta do aluno A.

Entrada
A entrada é composta por vários casos de teste, cada um correspondendo a uma simulação de eleição.

A primeira linha de um caso de teste contém apenas um inteiro, n (1 <= n <= 100), indicando a quantidade de alunos da escola (identificados por inteiros de 1 a n) e, por consequência, a quantidade de cédulas preenchidas. A seguir há n linhas, cada uma correspondendo a uma cédula processada.

Cada linha contém n inteiros, onde o j-ésimo inteiro da i-ésima linha é igual a 1, caso o j-ésimo quadrado da cédula i esteja preenchido (ou seja, o aluno de identificador i votou no aluno de identificador j); ou é igual a 0, caso contrário (o aluno de identificador i não votou no aluno de identificador j).

A entrada termina quando n = 0.

Saída
Para cada caso de teste, seu programa deve imprimir uma linha, contendo apenas um inteiro, correspondente à quantidade de votos recebidos pelo vencedor da eleição.

Exemplo

Entrada:

```
3
1 0 1
0 1 1
1 0 1
5
1 1 1 0 0
1 1 0 1 1
1 0 1 0 1
0 1 0 1 0
0 1 1 1 1
3
1 0 0
0 1 0
0 0 1
0
```

Saída:

```
3
4
1
```

Desafio 4 – Placar do Augusto

Prof. Augusto Jr. Mendes da Rede Laureate está muito preocupado com a queda do nível de atenção de seus estudantes. Ele já tentou várias técnicas mundialmente conhecidas para incentivar os alunos a prestar atenção nas suas aulas e fazer as tarefas que ele passa para a turma. Em sua última tentativa, lançou mão de diversas técnicas de metodologia ativa, mas todas se mostraram sem sucesso.

Dentre os principais recursos utilizados houve até pequenos casos de propina, tais como: nota para os alunos mais participativos, ofereceu chocolates aos que resolvessem os desafios de programação, levou seu karaokê e cantava nas aulas. Ameaçou até raspar o cabelo e as sobrancelhas caso os alunos programassem um jogo de tabuleiro na aula de vetores e matrizes em C…

Como tais medidas não levaram a uma melhora no comparecimento às aulas (a ideia do karaokê, inclusive, mostrou-se bastante infeliz... na segunda aula com karaokê a turma reduziu-se a um aluno -- que tinha problemas auditivos) ele teve uma brilhante ideia: faria uma competição entre os alunos.

Prof. Augusto passou um conjunto de problemas aos alunos, e deu um mês para que eles os resolvessem. No final do mês os alunos mandaram o número de problemas resolvidos corretamente. A promessa do brilhante didata era reprovar sumariamente o último colocado da competição, ou seja, sem que ele tivesse a oportunidade de fazer a prova N2 ou a Sub.

Os alunos seriam ordenados conforme o número de problemas resolvidos, com empates resolvidos de acordo com a ordem alfabética dos nomes (não há homônimos na turma).

Isso fez com que alunos com nomes iniciados nas últimas letras do alfabeto se esforçassem muito nas tarefas, e não compartilhassem suas soluções com colegas (especialmente aqueles cujos nomes começassem com letras anteriores).

Sua tarefa neste problema é escrever um programa que lê os resultados dos alunos do professor e imprima o nome do infeliz reprovado.

Entrada

A entrada é composta de diversas instâncias. A primeira linha de cada instância consiste em um inteiro n (1 <= n <= 100) indicando o número de alunos na competição.

Cada uma das n linhas seguintes contém o nome do aluno e o número de problemas resolvidos por ele. O nome consiste em uma sequência de letras [a-z] com no máximo 20 letras e cada time resolve entre 0 a 10 problemas.

A entrada termina com o valor 0 para a variável n.

Saída

Para cada instância, você deverá imprimir um identificador Instancia k, onde k é o número da instância atual. Na linha seguinte imprima o nome do infeliz reprovado.
Após cada instância imprima uma linha em branco.

Exemplo

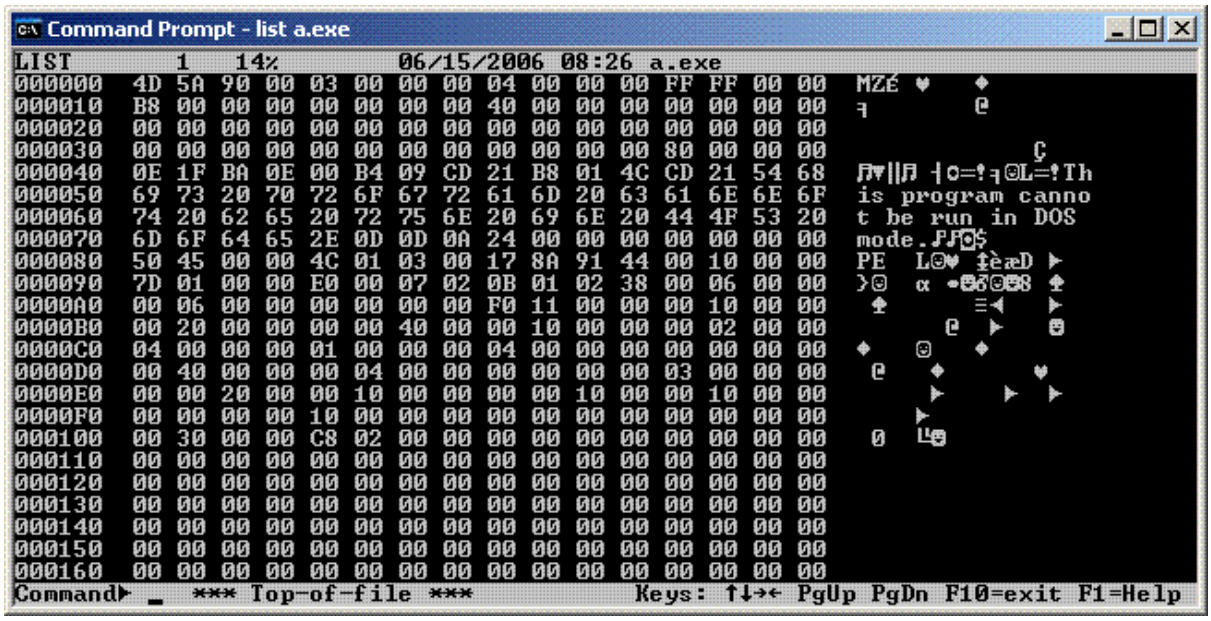
Entrada:
4
cardonha 9
infelizreprovado 3
marcel 9
infelizaprovado 3

Saída:
Instancia 1
infelizreprovado

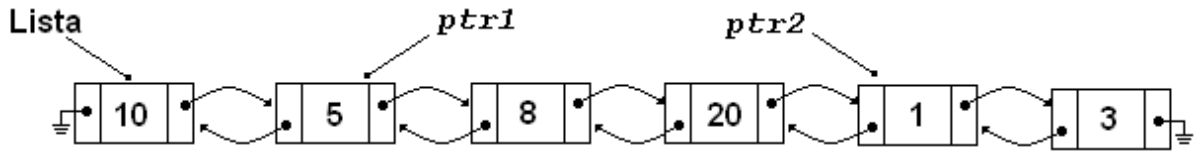
Desafio 5 – Ponteiros do Kakuga

Professor Kakugawa é um jovem e ambicioso professor de Estrutura de Dados (ED). Ao longo do tempo em que leciona a disciplina, percebeu erros recorrentes entre os alunos ao longo dos diversos semestres. Com o intuito de melhorar o processo de aprendizagem dos alunos, resolveu criar ferramentas que permitissem um acompanhamento mais pessoal. Afinal, a cada semestre que passa, a quantidade de alunos aumenta e fica cada vez mais difícil dar a atenção e o retorno merecido.

O professor Kakuga, como é carinhosamente denominado pelos seus alunos, é muito ocupado e não está conseguindo tempo para implementar uma ferramenta e por isso pediu a sua ajuda.



A ferramenta que o professor deseja implementar é um identificador de vazamento de memória durante a remoção de elementos de uma lista duplamente encadeada. No momento atual, o professor pede que os elementos removidos da lista duplamente encadeada sejam liberados da memória. Por enquanto, Kakuga deseja apenas que seja implementada a ferramenta que obtém o *dump* de memória atual e mostra como a lista ficará após a remoção de alguns elementos, permitindo ao aluno comparar como o seu programa se comportou.



Como exercício da disciplina de ED, o professor pediu o método da remoção que elimina da lista duplamente encadeada um conjunto de nós. São dados dois ponteiros *PTR1* e *PTR2* pertencentes a uma lista duplamente encadeada. É garantido que:

- Estes ponteiros existem e não são valores nulos (*NULL*);
- Estes ponteiros podem estar em qualquer posição da lista;
- Há um caminho entre *PTR1* e *PTR2* na lista;
- *PTR1* vem "antes" de *PTR2* na lista;
- Pode existir qualquer quantidade de nós antes de *PTR1*, depois de *PTR2* e entre *PTR1* e *PTR2* na lista.

Todos os nós entre *PTR1* e *PTR2*, inclusive *PTR1* e *PTR2*, devem ser removidos da lista duplamente encadeada, e a memória que usam deve ser liberada. Para auxiliar, busque pela função *free* (Saiba mais: https://www.ime.usp.br/~pf/algoritmos/aulas/aloca.html).

Como praxe, os alunos confundem os ponteiros e perdem referências, atualizam os ponteiros do nó anterior e próximo de forma incorreta, causando um verdadeiro caos na memória dos programas.

Para permitir que os alunos consigam identificar onde estão errando, e até confirmar se estão corretos, a ferramenta que você desenvolver receberá um *dump* da memória do programa do aluno contendo os nós apontados por *PTR1* e *PTR2* e outros nós, que podem ou não fazer parte da lista duplamente encadeada destes ponteiros, e determinar como a lista deverá ficar após a remoção, bem como quais elementos serão liberados da memória.

Entrada
A entrada contém o *dump* de memória do programa de um aluno. A entrada possui diversas linhas, e cada linha descreve um nó. Cada linha possui 3 números inteiros em formato hexadecimal representando, respectivamente, o endereço do nó, o endereço do nó anterior e do nó do próximo elemento. A entrada finaliza com a sequência de 3 zeros: 0 0 0.

O endereço 0 representa *NULL*. As duas primeiras linhas do caso de teste representam os nós apontados por *PTR1* e *PTR2*, nesta ordem.

Nem todos os ponteiros fornecidos na entrada precisam, necessariamente, fazer parte da lista duplamente encadeada. Além disso, o *dump* pode não conter a lista completa, ou seja, o nó mais anterior a *PTR1* não é, necessariamente, o primeiro elemento da lista (com o endereço anterior apontando para *NULL*), bem como o nó mais posterior a *PTR2* não é, necessariamente, o último nó da lista (com o endereço de próximo apontando para *NULL*). Entretanto, é garantido que existe pelo menos 1 nó anterior a *PTR1* e 1 nó posterior a *PTR2*.

Saída
A saída deverá conter diversas linhas, contendo o estado final da lista duplamente encadeada, i.e, após a remoção dos nós entre *PTR1* e *PTR2*. Os nós deverão ser ser impressos na ordem da lista encadeada: o primeiro nó impresso deverá ser o nó fornecido mais anterior de *PTR1* na lista, e o último nó impresso deverá ser o nó fornecido mais posterior a *PTR2* na lista.

Após a impressão da lista, uma linha deverá ser pulada e devem ser impressos os endereços de todos os nós removidos, na ordem em que apareciam na lista encadeada originalmente. Para entender melhor a saída, verifique os exemplos abaixo.

Exemplos

Entrada:

```
a 9 b
c b d
1 0 2
2 1 3
3 2 4
4 3 5
5 4 6
6 5 7
7 6 8
8 7 9
9 8 a
b a c
d c e
0 0 0
```

Saída:

```
1 0 2
2 1 3
3 2 4
4 3 5
5 4 6
6 5 7
7 6 8
8 7 9
9 8 d
d 9 e

a
b
c
```

Entrada:

```
a 9 b
c b d
2 1 3
fd fc fb
d c e
ff fb c
e d 10
10 e 20
1 0 2
fe fa fc
7 6 8
fb fd ff
```

8 7 9
fc fe fd
9 8 a
b a fa
fa b fe
0 0 0

Saída:
7 6 8
8 7 9
9 8 d
d 9 e
e d 10
10 e 20

a
b
fa
fe
fc
fd
fb
ff
c

Desafio 6 – Números Perfeitos na Matemática

Em Matemática, há um conceito muito interessante: os chamados números perfeitos. Um número natural é aquele cuja a soma de todos os seus divisores (excluindo ele próprio), é igual ao próprio número. Por exemplo: 28 = 1 + 2 + 4 + 7 + 14.

A partir desta classe de números, a professora Maricélia solicitou à você que elaborasse um programa sobre o assunto.

Entrada
A entrada consiste em números naturais: a e b. Sabe-se que a condição sempre será satisfeita: a<b. A partir destes 2 valores (incluindo a e b), o programa deverá identificar quais são números perfeitos. A leitura finaliza com: 0 0.

Saída
O programa deverá informar quantos números perfeitos foram identificados e na sequência, linha a linha, informar cada um desses números. Por último, finalize com uma linha em branco.

Exemplo

Entrada:
3 10
5 1000
0 0

Saída:
1
6

3
6
28
496

Desafio 7 - Operação Anticorrupção

O juiz Jorge Morro desconfia que dois deputados que passeiam todos os dias por um parque são na verdade desonestos e estão envolvidos no caso de corrupção da Oleobras. O parque é plano, de formato retangular, e estreitas faixas de grama o dividem em quadrados do mesmo tamanho, formando uma grade de N por M quadrados.

Os dois deputados têm um comportamento curioso e suspeito em seu passeio: após encontrarem-se, conversam durante um minuto, andam mudando rapidamente de lugar, passando a ocupar um novo quadrado do parque, conversam mais um minuto, andam novamente (mudando de quadrado), conversam mais um minuto, e assim sucessivamente. A cada minuto escolhem uma direção (Norte, Sul, Leste ou Oeste) e andam até o quadrado imediatamente vizinho na direção escolhida.

O juiz, suspeitando da ação, solicitou que a polícia federal procurasse interceptar as conversas entre os deputados, instalando um pequeno microfone multi-direcional em um dos quadrados do parque. O microfone é capaz de captar conversas realizadas no quadrado onde está instalado e em todos os quadrados imediatamente vizinhos.

Os dois deputados sempre iniciam o passeio pela entrada do parque que está no quadrado de coordenadas (0,0).

Dadas as coordenadas do microfone e a sequência de movimentos que os dois deputados realizam durante o passeio no parque, seu programa deve determinar quantos minutos de conversa foram captados pelo microfone.

Entrada
A primeira linha contém dois inteiros N e M que indicam respectivamente o número de linhas e o número de colunas do parque ($0 \leq N \leq 1000000$ e $0 \leq M \leq 1000000$).

A segunda linha contém dois inteiros X e Y que indicam a coordenada do microfone em termos de linhas e colunas ($0 \leq X \leq N$ e $0 \leq Y \leq M$).

A terceira linha contém um inteiro K, indicando o número de quadrados pelos quais os dois deputados passearam.

A quarta linha contém K inteiros, entre 1, 2, 3 e 4, que indicam a rota tomada pelos dois deputados durante o passeio; cada inteiro indica a direção tomada ao final de um minuto de conversa, representando da seguinte forma:
1 – Norte
2 – Sul
3 – Leste
4 – Oeste

Saída

Seu programa deve imprimir uma única linha contendo um inteiro, indicando o número de minutos de conversação captados pelo microfone.

Exemplos

Entrada
10 10
2 2
3
3 3 3

Saída
0

Entrada
5 5
0 1
3
3 1 3

Saída
3

Entrada
20 20
3 2
8
1 1 3 3 1 1 2 4

Saída
6

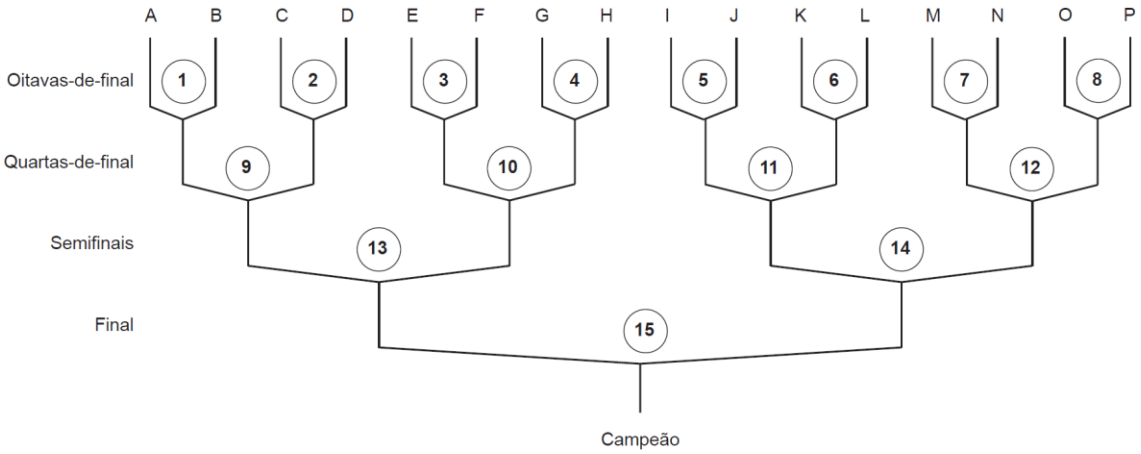
Desafio 8 - Copa do Mundo de Futebol

A Copa do Mundo FIFA é uma competição internacional de futebol que ocorre a cada quatro anos, sendo aberta as federações de países associados a FIFA (Federação Internacional de Futebol).

No Brasil a copa do mundo é um acontecimento singular, uma vez que o país vive o evento, onde o assunto entre a população são os jogos da competição. Mas quando a seleção brasileira enfrenta um adversário, outro fenômeno ocorre, o país suspende suas atividades em clima de festa para torcer pelo time canarinho conquistar mais um título mundial.

O torneio, atualmente, utiliza o um modelo que contém 32 seleções na fase de grupos, onde as seleções se enfrentam entre si dentro do seu grupo e os dois melhores são classificados para a fase final.

A fase final contém dezesseis equipes que realizam jogo único e eliminatório, ou seja, caso uma equipe perca o jogo está eliminada da competição. Desta maneira, esta etapa é composta de quinze jogos eliminatórios até resultar a equipe campeã conforme a ilustração a seguir:



A figura considera a Fase Final da copa do mundo, onde as dezesseis equipes finalistas são representadas pelas 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 da copa do mundo.

Entrada

A 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 $M \neq N$).

Saída

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

Exemplos

Entrada	Saída
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	F

Entrada	Saída
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	A

Desafio 9 – Calculadora Arbitraria

Sabemos que os números são representados de forma finita num computador, isto é, nunca poderíamos representar o número PI de forma exata numericamente, já que por ser um número irracional, não existe uma maneira de escrevê-lo em forma de fração. Do mesmo jeito, também não teríamos como guardar todas as casas decimais que representam PI, já que é infinito e o espaço em memória é finito, por maior que seja a quantidade de Gigabytes à sua disposição.

Além dos números irracionais, são frequentes os problemas nas áreas de ciências exatas onde é necessário efetuar cálculos com números inteiros com mais de 200 casas decimais. Um exemplo imediato é na área de criptografia, uma subárea da ciência da computação que estuda formas eficientes de proteger dados.

Para a nossa sorte, existem meios que nos permitem trabalhar com números dessa ordem com uma precisão muito além do que o tipo de dado inteiro (int) ou real (float/double) nos permite tratar. É neste ponto em que chegamos à computação de precisão arbitrária, ou seja, com números tão grandes quanto se queira, ainda que finitos.

Sua tarefa, neste projeto, será implementar uma calculadora de precisão arbitrária escrita em C, utilizando apenas a biblioteca padrão da linguagem. Faz parte do processo de desenvolvimento pesquisar métodos de representação de números inteiros de tamanho arbitrário, já que os tipos de dados oferecidos em C não dão conta do trabalho. Note que a calculadora só trabalhará com números inteiros, excluindo representações de ponto flutuante, facilitando muito o projeto.

Após a compreensão, por parte de todos do grupo, em como representar um número de tamanho arbitrário, será necessário elaborar uma solução algorítmica para tratar as operações básicas da aritmética com estes números. Num primeiro momento, seu projeto deverá contemplar apenas duas operações básicas: soma e multiplicação.

A seguir, alguns exemplos envolvendo cálculos com precisão arbitrária:

n1 = 6541695146517687468791
n2 = 8716544511000000000154165
n3 = 871869716987198674356245645786195673945687394568731569687356193875 6

n1 + n2 = 8723086206146517687622956

n1 * n2 = 57020976922014089471512175288463899288626164515

n1 * n1 = 427937753899730685197382226923152325990016 81

n2 * n2 = 759781482122442291236875721690766300000237668472 25

n3 * n3 = 760156803399337912667673970686422772330633210043901355700717961428842

03586431336379230693676327778638859394305473489489195141494827536

Vale lembrar que não é permitido o uso de qualquer biblioteca disponível para tratar cálculos de precisão arbitrária. Todos os algoritmos, além da teoria, devem ser desenvolvidos pelo grupo.

Tarefa

Dado os números e as operações, seu programa deve informar o valor final da expressão aritmética.

Entrada

A entrada contém, na primeira linha, a quantidade de números da expressão aritmética. As próximas linhas contêm os números intercalados pelas operações a serem realizadas. Vale ressaltar que a ordem que as operações aparecem, é a ordem que elas serão realizadas. Cada número tem no máximo 200 dígitos.

Saída

A saída será o valor da expressão aritmética. Caso o valor da expressão ultrapasse os 200 dígitos, o número deve ser arredondado.

Exemplo

Entrada
2 6541695146517687468791 + 871654451100000000154165
Saída
872308620614651768762295 6