Prova-04-sub

Prof. Msc. Elias Batista Ferreira
Prof. Dr. Gustavo Teodoro Laureano
Profa. Dra. Luciana Berretta
Prof. Dr. Thierson Rosa Couto

Sumário

1 Rede de Contatos no Twitter (+++)

 (+++)

Um determinado pesquisador precisa representar em uma tabela informações sobre relacionamentos entre um conjunto $\mathscr{U} = \{u_1, u_2, \ldots u_n\}$ de usuários do Twitter. Para isso, ele quer guardar para cada par $(u_1, u_2), u_1, u_2 \in \mathscr{U}$ as seguintes informações, quando existirem:

- O número de *likes* que u_1 fez em *tweets* escritos por u_2.
- O número de *retweets* que u_1 fez em fez em *tweets* escritos por u_2.
- O número de *menções* que u_1 fez em *tweets* escritos por u_2.

Par isso, o pesquisador quer criar uma matriz quadrada $n \times n$ que permita armazenar para cada par u_i, u_j, $1 \leq i, j \leq n$ as informações acima caso elas existam. Ele pensou em criar uma matriz de *structs* onde cada *structs* têm três campos correspondentes aos totais para cada um dos números de interações descritos acima. O problema dessa representação é que o pesquisador sabe que para um grande número de pares de usuários não existe nenhum tipo de interação entre eles, ou seja, os totais para os três tipos de interação seriam iguais a zero. Além disso, há vários casos em que u_1 interage com u_2, mas u_2 não interagem com u_1. Armazenar todas as $n \times n$ *structs* na tabela é, portanto, um desperdício de memória e n pode ser muito grande impossibilitando o programa de funcionar. O pesquisador quer que você faça um programa em que a tabela $n \times n$ seja uma tabela de ponteiro para as *structs* que possuem os três campos comentados anteriormente e que aloque as structs sob demanda, somente quando forem necessárias. O programa deve ler os dados dos relacionamentos entre usuários e imprimir para cada usuário o total de *likes*, o total de *retweets* e o total de *menções* que ele fez a *tweets* de outros usuários. Além disso, é importante também saber a quantidade de *slots* vazios na tabela os valores médios de *likes*, *retweets* e *menções* por usuário.

Entrada

A primeira linha da entrada é constituída por um único inteiro positivo $N(N \leq 1000)$, o qual corresponde à dimensão da matriz de relacionamentos. A segunda linha contém outro número inteiro M, $1 \leq M \leq N^2$ que corresponde ao número de pares não nulos da matriz. Em seguida há M linhas, cada uma com o seguinte formato: u_1 u_2 *num_likes num_retweets num_menções*, onde os campos estão separados entre si por um espaço. O primeiro campo u_1 corresponde ao usuário que fez as interações. O segundo campo (u_2) corresponde ao usuário que recebe as interações. Os demais três campos correspondem a, respectivamente, o número de likes, o número de retweets e o número de menções que u_1 fez em tweets de u_2.

Saída

O programa deve imprimir uma linha para cada usuário contendo as seguintes informações: Usuario u_i - num. likes feitos: x, num. retweets feitos: y e num. mencoes feitos: z. Os valores de x, y e z correspondem ao totais de likes, retweets e menções, respectivamente, feitos pelo usuário u_i. Os usuários devem ser impressos na ordem crescente do seu número (ordem das linhas da tabela).

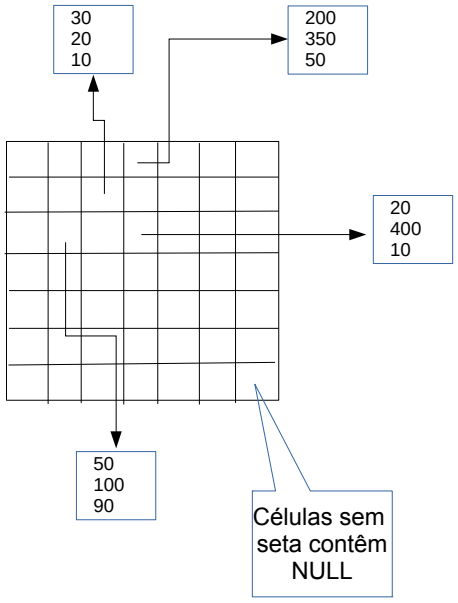
Observações

1. Os compiladores da linguagem C não inicializam os campos de uma tabela. Você deve garantir que antes da leitura todos as células da tabela tenham NULL como valor de ponteiro.
2. Ao final do programa, todo espaço alocado dinamicamente deve ser liberado
3. Uso da função malloc() : x = (Tipo_do_x*) malloc(Num_bytes_tipo_x * quantidade_elem). A função malloc() retorna NULL caso não consiga alocar epaço. Seu programa deve verificar se foi possível alocar espaço e terminar o programa em caso contrário.
4. Deve ser feito #include<stdlib.h> para usar a função malloc().

5. A função exit(1) encerra o programa.
6. A função free(x) libera a área de memória cujo endereço está em x.
7. dada uma matriz mat, para acessar o campo num_likes da struct cujo endereço está em mat[i][j]: (*mat[i][j]).num_likes. Exemplo scanf("%d", &((*mat[i][j]).num_likes)).

Exemplo

```
Entrada:
7
4
0 3 200 350 50
1 2 30 20 10
2 1 50 100 90
2 3 20 400 10
```




```
Saída:
Usuario 0 - num. likes: 200, num. retweets: 350 e num. mencoes: 50
Usuario 1 - num. likes: 30, num. retweets: 20 e num. mencoes: 10
Usuario 2 - num. likes: 70, num. retweets: 500 e num. mencoes: 100
Slots vazios: 45
Media de likes por usuario: 100.00
Media de retweets por usuario: 290.00
Media de mencoes por usuario: 53.33
```

2 Loteria (+++)

 (+++)

A Loteria é um jogo que paga um prêmio em dinheiro para o apostador que conseguir acertar os 6 números sorteados. Ainda é possível ganhar prêmios ao acertar 4 ou 5 números dentre os 60 disponíveis no volante de apostas. Para isso, você deve *marcar* 6 números do **volante**. Você poderá fazer quantas apostas quiser, ou seja, poderá jogar quantos volantes necessitar. Os números estão entre 1 e 60.

Faça um programa que receba os jogos de um apostador, em seguida, leia o resultado da loteria e verifique se o apostador acertou os números sorteados. Se o apostador acertou 4, 5 ou 6 números é necessário emitir um aviso reportando o fato.

É obrigatório utilizar estrutura para armazenar os números apostados e o resultado .

```
typedef struct {
   int numJogo;
   int numero[6];
} CARTELA;
```

Entrada

A entrada contém vários casos de teste. A primeira linha de cada caso de teste contém um inteiro $N(1 \leq N \leq 10^3)$, indicando a quantidade de apostas do jogador. As N linhas seguintes contém o número do jogo e 6 números correspondentes aos palpite do jogador.

Em seguida, deverá ter um linha para ler o número do concurso e os 6 números sorteados, que devem ser armazenados em outra estrutura.

*** Deve-se utilizar alocação dinâmica para reservar N espaços das apostas.**

Saída

Para cada entrada, deve-se verificar se o apostador acertou, no mínimo, 4 números e emitir a seguinte mensagem:

1. QUADRA jogo: a b c d: quando a apostador acertar 4 números.
2. QUINA jogo: a b c d e: quando a apostador acertar 5 números.
3. SENA jogo: a b c d e f: quando a apostador acertar 6 números.

Após analisar todas as apostas e constatar que o apostador não conseguiu acertar, no mínimo, 4 números, escreva a mensagem "NENHUMA CARTELA PREMIADA PARA O CONCURSO concurso".

Exemplos

Entrada
4 1 5 15 25 35 45 55 2 9 13 28 46 51 52 3 2 28 46 47 51 13 4 8 15 25 35 45 55 1050 9 13 28 46 51 52
Saída
SENA 2: 9 13 28 46 51 52 QUADRA 3: 28 46 51 13

Entrada
3 1 3 11 44 50 56 32 2 2 12 57 51 45 33 3 1 34 13 46 58 52 1051 5 15 36 47 53 60
Saída
NENHUMA CARTELA PREMIADA PARA O CONCURSO 1051