Prova 2
Prof. Hebert Coelho

Conteúdo

1 Formiga

 (+++)

Imagine um tabuleiro quadrado $N \times N$, que pode ser uma matriz $N \times N$. As casas "livres" são marcadas com 0 (zero) e as casas "bloqueadas" com -1 (menos um). As casas (1,1) e (N,N) estão livres. Explique como ajudar uma formiga $\bowtie$ que está na casa (1,1) a chegar à casa (N,N) que contém um doce $\bigoplus$ pelo menor caminho possível. Em cada passo, a formiga só pode se deslocar para uma casa livre que esteja à direita, à esquerda, acima ou abaixo da casa em que está.

Entrada: A primeira linha contém a dimensão N da matriz. As próximas N linhas contém os N elementos de cada linha da matriz.

Saída: Você deverá imprimir um valor inteiro com a quantidade de casas percorridas pela formiga utilizando o menor caminho, contando inclusive com a casa (0,0) e a casa (N,N).

Exemplo:
Entrada
10

$\bowtie$	0	0	0	0	-1	0	0	0	0
0	-1	0	-1	0	-1	-1	0	-1	0
0	-1	0	-1	0	0	-1	0	-1	0
0	-1	0	-1	-1	-1	-1	0	-1	0
0	-1	0	0	0	0	0	0	-1	0
0	-1	-1	-1	-1	-1	-1	-1	-1	0
0	-1	0	0	0	0	-1	0	0	0
0	-1	0	0	0	0	-1	0	0	-1
0	-1	-1	-1	-1	-1	-1	-1	0	0
0	0	0	0	0	0	0	0	0	$\bigoplus$

Saída:
19

2 Radix

 (+++)

Existe uma aplicação interessante para filas. Seja L uma lista linear simplemente encadeada composta de n chaves inteiras na base $b > 1$, neste exercício sempre o valor de $b = 10$. O problema consiste em ordenar a lista L. Para tal são necessários utilizar b filas em alocação encadeada (0 até $b-1$). O algoritmo efetua $d \leq 7$ iterações (d representa a quantidade de casas decimais). A primeira iteração verifica o dígito menos significativo da representação decimal. Se este for igual a k, a chave correspondente será inserida na fila F_k (implementar as filas com uso de alocação dinâmica e não com vetor). Ao terminar o percurso da tabela, esta se encontra distribuída pelas filas, que devem então ser concatenadas em sequência, isto é, F_0, depois F_1, F_2, etc. O processo deverá ser repetido levando-se em consideração o segundo digito, e assim por diante até que tenham sido feitas tantas distribuições quantos são os dígitos na chave de ordenação. Veja abaixo um exemplo dessa ordenação, onde $b = 10$ e $d = 2$.
Lista simplesmente encadeada: 19; 13; 05; 27; 01; 26; 31; 16; 02; 09; 11; 21; 60; 07;
Iteração 1: 1^a distribuição (unidade simples)
F_0 : 60
F_1 : 01, 31, 11, 21
F_2 : 02
F_3 : 13
F_4 :
F_5 : 05
F_6 : 26, 16
F_7 : 27, 07
F_8 :
F_9 : 19, 09
Lista simplesmente encadeada: 60; 01; 31; 11; 21; 02; 13; 05; 26; 16; 07; 27; 19; 09
Iteração 2: 2^a distribuição (dezena simples)
F_0 : 01, 02, 05, 07, 09
F_1 : 11, 13, 16, 19
F_2 : 21, 26, 27
F_3 : 31
F_4 :
F_5 :
F_6 : 60
F_7 :
F_8 :
F_9 :
Lista simplesmente encadeada: 01; 02; 05; 07; 09; 11; 13; 16; 19; 21; 26; 27; 31; 60

Entrada:
A primeira linha contém a quantidade de casos de teste m. As m linhas seguintes contém cada, a quantidade de dígitos $d > 0$, uma sequência de n valores inteiros positivos que representam uma lista L_i, onde $1 \leq i \leq m$ e ao final o valor -1 que finaliza cada lista.

Saída:
Seu programa deverá imprimir: m linhas cada uma contendo a lista L_i ordenada, $1 \leq i \leq m$.

Exemplos: Exemplo 1 é o mesmo exemplo da descrição do problema

Entrada 1:
1
2 19 13 05 27 01 26 31 16 02 09 11 21 60 07 -1

Saída 1:

01 02 05 07 09 11 13 16 19 21 26 27 31 60

Entrada 2:

2
3 191 132 053 274 015 266 317 168 029 091 112 213 604 075 -1
4 1000 2000 1500 2300 4800 8900 1234 1235 1765 1978 1975 1950 -1

Saída 2:

015 029 053 075 091 112 132 168 191 213 266 274 317 604
1000 1234 1235 1500 1765 1950 1975 1978 2000 2300 4800 8900

3 Cálculo de Expressão em Notação Polonesa

 (++)

Muitos alunos que baixam apps de calculadoras mais avançadas tem dificuldade em gerar as expressões em notação polonesa (pós-fixa). Você deve implementar um programa que receba uma expressão em notação polonesa e calcule o resultado dessa expressão. A expressão será digitada com um "\n" (enter) entre cada operador ou operando. Os únicos valores possíveis na expressão serão valores inteiros com no máximo 10 digitos e os operadores adição (+), subtração (-), multiplicação (*) e Divisão (/). A estrutura de dados usada para armazenar os valores inteiros deverá ser implementada com alocação encadeada (dinâmica).
Exemplo 1 de expressões:
Notação infixa: (4 + (5 * 6)) - (2 + 7)
Notação pós-fixa: 4 5 6 * + 2 7 + -
Resultado das expressões acima 25

Exemplo 2 de expressões:
Notação infixa: 5 + (1 + 2) * 4 - 3
Notação pós-fixa: 5 1 2 + 4 * + 3 -
Resultado das expressões acima 14

Entrada:
A primeira linha contém um inteiro *N* com o número de linhas após a primeira. As próximas *N* linhas contém ou um valor inteiro ou um operador.

Saída:
Seu programa deve imprimir um valor inteiro com o resultado da expressão.
Exemplo:

Entrada:
9
40
5
6
*
+
2
7
+
-
Saída:
61

Obs.: A biblioteca "stdlib.h" contém uma função que converte uma string apontada por str em um valor inteiro, o prototipo da função é: int atoi(const char *str);
Caso o valor não seja um inteiro como "+", "-", "*" ou "/" a maioria das implementações de atoi retorna 0.

4 Cidades

 (+++)

Imagine n cidades numeradas de 0 até $n-1$ e interligadas por estradas de mão única. As ligações entre as cidades são representadas por uma matriz A definida da seguinte maneira: $A[x][y] = 1$ se existe a estrada da cidade x para a cidade y e $A[x][y] = 0$ caso contrário. A **distância** de uma cidade x a uma cidade y é o menor número de estradas que é preciso percorrer para ir de x a y. **Problema**: Escrever um programa que recebe a matriz A, o tamanho da matriz n e a cidade x e que determina a distância da cidade x para cada uma das outras cidades. Obs.: Se não for possível chegar da cidade x a uma cidade y representar a distância com o valor -1.

Entrada:
A primeira linha contém a dimensão n da matriz e a cidade $0 \leq x < n$. As n linhas seguintes contém cada uma sequência de n valores (0 ou 1) que representam cada, uma linha da matriz de ligações.

Saída:
Seu programa deverá imprimir as distâncias da cidade x a todas as outras cidades.

Exemplo:

Entrada:
6 3
0 1 0 0 0 0
0 0 1 0 0 0
0 0 0 0 1 0
0 0 1 0 1 0
1 0 0 0 0 0
0 1 0 0 0 0
Saída:
2 3 1 0 1 -1

5 Lista Linear

 (+++)

A empresa Coelho Cosmic Space (CCS), esta na competição para enviar pessoas fora da orbita terrestre, os dispositivos de suas aeronaves são todos programáveis em linguagem C, assim como os códigos seguros desenvolvidos pela NASA. Você foi designado para implementar a inserção e remoção em uma Lista Linear Circular Ordenada (sem nó cabeça) que utilize no máximo 10 celulas, usando alocação encadeada (ponteiros). A estrutura da Lista é:

```
typedef struct no_Lista{
int id; //identificador do dispositivo
int p; //prioridade do dispositivo
struct no_Lista *proximo; //ponteiro para próximo elemento
} NO;
```

A Lista Linear deverá ser ordenada pelo id do dispositivo. Não existem dois dispositivos com o mesmo identificador. Assim, na inserção você não deverá inserir um dispositivo com id que já esteja na lista.

Entrada: A primeira linha contém um inteiro *N* com o número de linhas após a primeira. As próximas *N* linhas são de dois tipos na entrada, uma para inserção com 3 elementos e um para remoção com 2 elementos. Todas as linhas da inserção contém 3 elementos "*X*" "*Y*" "*Z*". O primeiro elemento *X* é um inteiro com valor 1 "inserção". O segundo elemento *Y* é um inteiro indicando o identificador (id) do dispositivo. O terceiro elemento *Z* é um inteiro indicando a prioridade do dispositivo. Todas as linhas da remoção contém 2 elementos "*X*" "*Y*". O primeiro elemento *X* é um inteiro com valor 2 "remoção". O segundo elemento *Y* é um inteiro indicando o identificador (id) do dispositivo.

Saída: Seu programa deve imprimir as mensagens:
Overflow - quando não houver mais memória e a inserção for chamada;
Underflow - quando a lista estiver vazia e a remoção for chamada;
Valor inteiro com a posição do nó que já existe na estrutura - quando a inserção não puder ocorrer devido ao identificador do processo;
RProID - quando a remoção não puder ocorrer devido ao id do processo inexistente;
Valor inteiro com a posição do nó removido na estrutura - Quando a remoção acontecer;

Overflow e Underflow são prioritários com relação as outras saídas.

Exemplo:

Entrada

```
10
 2  10
 1  11  5
 1  11  6
 1  10  6
 1   3  6
 1   4  7
 1  10  5
 2   4
 2   7
 2  10
```

Saída:

```
Underflow
1
3
2
RProID
2
```