Estrada de Wakanda

Para melhorar a integração com os países vizinhos, o Rei T'Challa de Wakanda decidiu que uma nova estrada será construída cruzando o país, da fronteira oeste à fronteira leste. O formato da estrada é uma única reta, que passará pelo centro de algumas cidades.

O Rei T'Challa também decidiu que a construção será paga pelo Tesouro Real, mas cada cidade pela qual a estrada passar será responsável pela manutenção do trecho da estrada que constitui a vizinhança da estrada para aquela cidade.

Atenção:

A vizinhança da estrada de uma cidade A é definida como todos os pontos da estrada que são mais próximos do centro da cidade A do que do centro de qualquer outra cidade.

Dados o comprimento total da estrada, de fronteira a fronteira, e as distâncias da fronteira oeste até os centros de cada cidade ao longo da nova estrada, escreva um conjunto de funções para determinar qual a menor vizinhança de estrada entre as cidades pelas quais a estrada vai passar, qual a cidade que tem a menor vizinhança e para inserir dados lidos em uma lista encadeada.

Entrada

A primeira linha da entrada (nome do arquivo texto passado como parâmetro) contém um inteiro T, o comprimento total da estrada. A segunda linha contém um inteiro N, o número de cidades pelas quais a estrada vai passar. Cada uma das N linhas seguintes contém um inteiro Xi e uma string Si, indicando a distância da fronteira oeste até o centro da cidade i e o nome da cidade i. Não há cidades nas fronteiras e cada centro de cidade tem uma localização distinta.

Saída

A parte do programa que você deve implementar consiste de arquivo **cidades.c** contendo a implementação das funções cujos protótipos estão definidos em cidades.h, quais sejam:

```
Estrada *getEstrada(const char *nomeArquivo);             // Inicializa cidades no TAD Cidade.
double calcularMenorVizinhanca(const char *nomeArquivo); // Retorna a menor vizinhança.
char *cidadeMenorVizinhanca(const char *nomeArquivo);     // Retorna a cidade que tem menor vizinhança.
```

Restrições

- $3 \leq T \leq 10^6$ e $2 \leq N \leq 10^4$
- $0 < Xi < T$, para $1 \leq i \leq N$
- $Xi \neq Xj$, para todo par $1 \leq i, j \leq N$.

Atenção:

As restrições devem ser verificadas no códifo-fonte e devem implicar no retorno de ponteiro nulo para a função getEstrada.

Exemplo

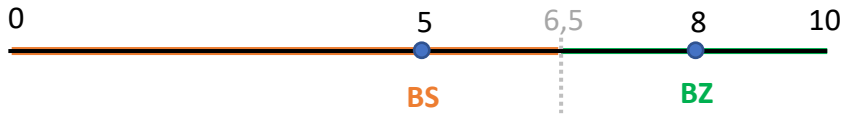
A seguir é apresentado um exemplo de arquivo de entrada, cujos dados permitem inferir que 3.50 é a menor vizinhança dentre as cidades indicadas e está associada à cidade Birnin Zana.

Entrada (arquivo teste01.txt)

```
10
2
8 Birnin Zana
5 Birnin S'Yan
```

A vizinhança da estrada de uma cidade A é definida como todos os pontos da estrada que são mais próximos do centro da cidade A do que do centro de qualquer outra cidade.

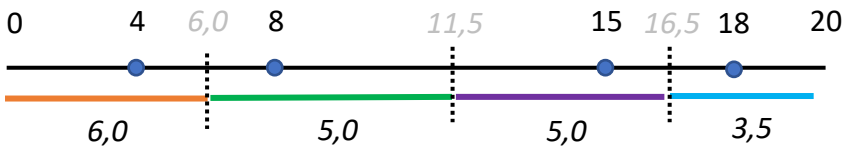
Nesse caso, a vizinhança de Birnin Zana tem 6,5 km e a de Birnin S'Yan tem 3,5 km, conforme



Par ao exemplo a seguir, as vizinhanças de Birnin S'Yan, Birnin Zana, K'pur e L'pur são, respetivamente, iguais a 6,0; 5,5; 5,0 e 3,5.

Entrada (arquivo teste02.txt)

```
20
4
8 Birnin Zana
4 Birnin S'Yan
15 K'pur
18 L'pur
```



É altamente recomendável que sejam elaborados outros exemplos para, efetivamente, confirmar a correção das funções implementadas.

Arquivo cidades.h

Necessariamente, o arquivo cidades.h deve ter o seguinte conteúdo e **não pode ser modificado**.

```c
#ifndef CIDADES_H
#define CIDADES_H

typedef struct {
    char Nome[256];     // Nome do cidade
    int Posicao;        // Posição da cidade
} Cidade;

typedef struct {
    int N;              // Número de cidades
    int T;              // Comprimento da estrada
    Cidade *C;          // Vetor de cidades
} Estrada;

Estrada *getEstrada(const char *nomeArquivo);               // Inicializa cidades no TAD indicado acima.
double calcularMenorVizinhanca(const char *nomeArquivo);    // Retorna a menor vizinhança.
char *cidadeMenorVizinhanca(const char *nomeArquivo);   // Retorna a cidade que tem menor vizinhança.

#endif
```

Observações

1. Você deverá entregar apenas o arquivo **cidades.c**, contendo as implementações das funções cujos protótipos foram definidos em cidades.h e sem incluir a função main(). Nesse arquivo, você pode implementar funções adicionais para auxiliar no desenvolvimento deste trabalho prático.

2. No arquivo **cidades.c**, podem ser implementadas outras funções acessórias, mas os protótipos dessas funções adicionais não devem ser incluídos em cidades.h. Ademais, caso ocorra erro na inicialização da estrada em `Estrada *getEstrada(const char *nomeArquivo)`, esta deverá retornar ponteiro **NULL**.

3. O código-fonte deve ser desenvolvido seguindo preceitos da linguagem C padrão, especialmente no que diz respeito à inclusão de bibliotecas, alocação de memória para ponteiros, diferenciação entre variáveis locais, etc.

4. A correção deste trabalho será automática a partir da instrução **gcc main.c cidades.c –o cidades.exe**, em que **cidades.c** é sua proposta de resolução para este problema. No caso, seu código deve ter compatibilidade de compilação com o gcc (na linguagem C padrão), em IDE's como DevC ou VSCode. Um exemplo de main.c é mostrado a seguir:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "cidades.h"

int main () {

    int Soma = 0;
    int NumTeste = 0;

    FILE *Resposta = fopen("Resultado.txt", "w");

    Estrada *T1 = getEstrada("Teste01.txt");
    double D1 = calcularMenorVizinhanca("Teste01.txt");
    char *C1 = cidadeMenorVizinhanca("Teste01.txt");

    if (T1->T == 10) Soma++;
    NumTeste++;
```

```
    if (T1->N == 2) Soma++;
    NumTeste++;

    if (D1 == 3.5) Soma++;
    NumTeste++;

    if (strcmp(C1, "Birnin Zana")==0) Soma++;
    NumTeste++;

    fprintf(Resposta, "\n\nATENÇÃO: Você acertou %d de %d itens. Logo, em 2.00 pontos, sua nota
    foi %.2f.\n", Soma, NumTeste, 2.0 * (float)Soma/(float)NumTeste);
}
```

5. Ademais, a não observância dos requisitos presentes neste documento pode implicar em decréscimo de nota.