Exercício-Programa 1: Rota de menor custo em mapas

Objetivo:

Desenvolver um programa para encontrar rotas de menor custo em um mapa. O programa deverá ser desenvolvido em C e deverá ser executável via linha de comando.

Entrada:

A chamada do programa será:

gps.exe <arquivomapa> <arquivorequisicoes> <arquivosaida>

Exemplo:
gps.exe mapa01.txt reqrota01.txt caminho01.txt

<arquivomapa>:
Conterá a representação do mapa através de um grafo, e terá o seguinte formato: A primeira linha conterá o número de vértices e o número de arestas do grafo – ambos inteiros. Cada uma das demais linhas conterá os campos de uma aresta:
- Nome da rua (string sem espaços e sem aspas, até 50 caracteres)
- Vértice inicial (inteiro)
- Vértice final (inteiro)
- Número inicial na rua (inteiro longo)
- Número final na rua (inteiro longo)
- Custo (double).

Os delimitadores de campos podem ser espaço ou tabulação. Assume-se que o grafo é simétrico, ou seja, não há ruas de mão única, e o custo de uma aresta é o mesmo em ambas as direções.

O exemplo abaixo ilustra um trecho de um arquivo com 30 vértices e 80 arestas:

```
30 88
B 0 1 0 80 8
B 1 0 80 0 8
A 0 2 0 60 6
A 2 0 60 0 6
B 1 3 80 150 7
B 3 1 150 80 7
D 1 4 0 60 6
D 4 1 60 0 6
...
```

<arquivorequisicoes>:
Conterá as requisições de rotas, e terá o seguinte formato: A primeira linha conterá o número de requisições. Cada uma das demais linhas conterá o endereço de origem e o endereço de destino:

- Nome da rua de origem (string sem espaços e sem aspas, até 50 caracteres)
- Número de origem (inteiro longo)
- Nome da rua de destino (string sem espaços e sem aspas, até 50 caracteres)
- Número de destino (inteiro longo)

Assume-se que os endereços sejam válidos, ou seja, que as ruas de origem e destino existam no mapa e que os números de origem e destino não ultrapassem os limites das ruas correspondentes.

Exemplo:

```
3
I 64 P 11
G 26 P 106
J 64 O 19
```

<arquivosaida>:
Conterá as rotas entre os endereços de origem e destino, e seus respectivos custos. A primeira linha conterá o número de requisições. As demais linhas conterão as rotas, divididas em bloco.
A linha inicial de cada bloco conterá:
- Nome da rua de origem (string sem espaços e sem aspas, até 50 caracteres)
- Número de origem (inteiro longo)
- Nome da rua de destino (string sem espaços e sem aspas, até 50 caracteres)
- Número de destino (inteiro longo)
- Custo mínimo (double)
- Número de ruas distintas do endereço de origem ao endereço de destino (inteiro)

As demais linhas do bloco conterão os nomes das ruas por onde a rota de menor custo deverá passar. Note que somente as ruas distintas no trajeto deverão ser exibidas (arestas consecutivas com mesmo nome de rua não devem ser listadas na rota final).
No exemplo abaixo, são apresentadas as rotas para as requisições do exemplo anterior.

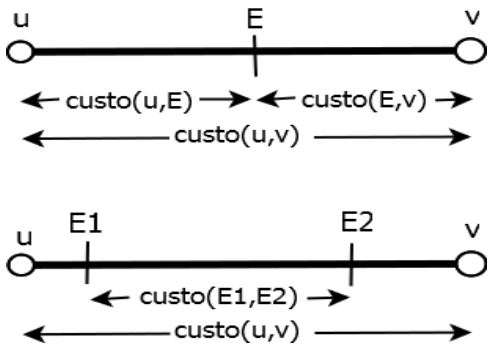
```
3
I 64 P 11 33.300000 5
I
G
M
O
P

G 26 P 106 28.800000 2
G
P

J 64 O 19 24.700000 3
J
```

M
O

Implementação

Para determinar o melhor caminho entre dois endereços, utiliza-se a idéia geral descrita abaixo.

Sejam E_o o endereço de origem e E_d o endereço de destino. Denotemos por (o_1, o_2) e (d_1, d_2) as arestas dos endereços E_o e E_d, respectivamente. Denote por $melhorcaminho(u, v)$ o caminho de menor custo de u a v, obtido pelo algoritmo de Dijkstra, e denote por $customin(u, v)$ o custo desse caminho. Consideramos ainda a função *custo* sobre uma aresta (vide diagrama abaixo): seja (u, v) uma aresta, e E, E_1, E_2 endereços localizados na aresta (u, v).
- $custo(u, v) = custo(v, u)$ é o custo da aresta (u, v);
- $custo(u, E) = custo(E, u)$ é o custo do segmento da aresta entre o endereço E e o vértice u;
- $custo(E_1, E_2)$ é o custo do segmento da aresta entre os endereços E_1 e E_2.



Para determinar o melhor caminho do endereço E_o ao endereço E_d, deverão ser testadas as alternativas abaixo:

-$E_o \to o_1 \to melhorcaminho(o_1, d_1) \to d_1 \to E_d$
-$E_o \to o_1 \to melhorcaminho(o_1, d_2) \to d_2 \to E_d$
-$E_o \to o_2 \to melhorcaminho(o_2, d_1) \to d_1 \to E_d$
-$E_o \to o_2 \to melhorcaminho(o_2, d_2) \to d_2 \to E_d$
-Se E_o e E_d estiverem sobre a mesma aresta, a quinta opção será $E_o \to E_d$.

Os custos desses caminhos serão, respectivamente:
$custo(E_o, o_1) + customin(o_1, d_1) + custo(d_1, E_d)$
$custo(E_o, o_1) + customin(o_1, d_2) + custo(d_2, E_d)$
$custo(E_o, o_2) + customin(o_2, d_1) + custo(d_1, E_d)$
$custo(E_o, o_2) + customin(o_2, d_2) + custo(d_2, E_d)$,
$custo(E_o, E_d)$.

O melhor caminho será o de menor custo dentre as possibilidades acima.

Entrega do trabalho:

O trabalho poderá ser individual ou em duplas.

Anexo a este enunciado, encontram-se os módulos grafosmapa.c, indheap.c, dicionario.c (e seus respectivos *headers* .h), que poderão ser utilizados para a implementação do programa final. O módulo dijkstramapa.c, será o módulo principal, e contém os protótipos das principais funções a serem implementadas.

Condições da entrega:

- Deverá ser entregue o módulo dijkstramapa.c renomeado na forma

  d<numerousp1>_<numerousp2>.c

  onde numerousp1 e numerousp2 correspondem aos números USP dos membros da dupla.

  Exemplo: d1234567_7654321.c

  Este módulo conterá as implementações das funções *Dijkstra*, *CarregaRequisicoes*, *ImprimeMelhorRota* e *main*.

  Quaisquer outras funções que o aluno desejar implementar deverão ser escritas dentro deste módulo. Não altere nenhum dos demais módulos listados acima.

- O código-fonte deverá ser compilável no gcc.

- O trabalho deverá ser enviado por email, em endereço a ser divulgado oportunamente, com o assunto ACH2024-EP1.

- O prazo para entrega é 25/05/2014.

- Dúvidas a respeito das especificações ou a respeito da implementação do trabalho serão sanadas até o dia 22/05/2014. Dúvidas encaminhadas após este prazo serão ignoradas.

- Além da correção do programa, será considerada a qualidade da documentação do código fonte.

- Evidência de plágio entre trabalhos sujeitará os alunos envolvidos à reprovação.