MC336 - PROJETO 3
Caixeiro Viajante

Escreva um programa em Haskell que resolve o problema do caixeiro viajante, ou seja, dado as coordenadas de n cidades, e a cidade de origem, determine o caminho que sai da cidade de origem, visita as outras $n-1$ cidades e retorna à origem que possui a menor distancia percorrida total.

Para a solução deste problema voce terá que gerar todas as permutações das $n-1$ cidades que não a origem, e verificar a distancia percorrida total para cada uma destas alternativas.

Os dados de entrada estão no seguinte formato:

```
origem
cidade1  coordx  coordy
cidade2  coordx  coordy
...
```

onde um ou mais brancos separam os dados de cada linha (e portanto perfeito para usar a função `words`). Cada cidade tem duas coordenadas (x e y) e a distancia entre cidades deve ser calculado como a distancia euclidiana entre pontos em 2 dimensões.

Imprima o custo total do menor caminho arredondado para um inteiro (mas só arredonde na hora da impressão - faça os cálculos de distancia com `Double`) e na linha seguinte o caminho na forma

```
[origem, cidade1, cidade2, ... cidaden-1, origem]
```

Note que há sempre 2 caminhos com menor custo. Se $[a, b, c, d, a]$ é o caminho de menor distancia percorrida, então o reverso $[a, d, c, b, a]$ também o é. Destas duas alternativas o seu programa deve imprimir aquele caminho onde o nome da 2a cidade (o b no primeiro caso) é lexicograficamente menor que o nome da penúltima cidade (o d).

O susy rodara seu programa como:

```
runghc ex3.hs < arqx.in
```

ou seja:

- o nome da função principal do seu programa deve ser `main`
- a leitura é como se fosse do teclado (ou stdin).
- a saída deve ser pelo comando `print`