CS1101C
AY2001/2 Semester 2 Practical Examination

Write a program to perform the following 3 tasks.

Task 1
Write two functions to:

a) retrieve information about the existing bus system from a user-specified file, and

b) print out the retrieved information as a table such that entry (*i*, *j*) in the table is the number of buses going from bus stop *i* to stop *j*.

The function names for tasks 1a and 1b are **buildTable** and **printTable** respectively. You can use any appropriate parameters and return types for these two functions.

The user-specified file is formatted in the following manner

`<source stop> <destination stop> <bus service number>`

You may make the following assumptions:

- Input files contain valid data in the correct format.
- Each source, destination, and the bus service number is defined using an integer ranging from 0 to 7 (inclusive).
- There are at most 8 bus services at any given bus stop.
- Some bus stops may not have any bus service.
- Bus services are bi-directional. For example, bus 2 travelling from stop 0 to stop 1 is assumed to travel from stop 1 to stop 0 also. Sometimes, both directions (0 1 2) and (1 0 2) are provided in the input; sometimes, only one direction is provided.
- The **buildTable** and **printTable** functions may invoke other additional functions that you may write in order to retrieve and print information about the existing bus system.

A sample input file, **pe.in**, is shown below. The first line contains an integer indicating the number of data lines following it.

```
10
0 1 2
0 2 3
0 3 4
1 3 4
1 0 5
2 0 3
1 3 6
0 3 1
0 3 6
3 1 7
```

The corresponding table printed by **printTable** given the above data is as follows:

```
0   2   1   3   0   0   0   0
2   0   0   3   0   0   0   0
1   0   0   0   0   0   0   0
3   3   0   0   0   0   0   0
0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0
```

(Hint: Use a 3-dimensional array to store the information retrieved in the **buildTable** function.)

Task 2

Write a function **handleQueries** to answer user queries about bus services serving user-specified bus stops. You can use any appropriate parameters and return type for this function. This function must perform the following tasks.

- Ask user for a bus stop number. If the number is invalid, for example, stop numbers greater than 7, then the function should print the error message

  `Invalid stop number: <stop number>`

  where <stop number> is the invalid value that the user entered.

- Print out the bus service numbers, in ascending order, that visit the stop number. If there are no bus service serving the user-specified stop number, then the function should print the message

  `No buses found!`

- Ask user whether he or she wants to input another query.

You may make the following assumptions:

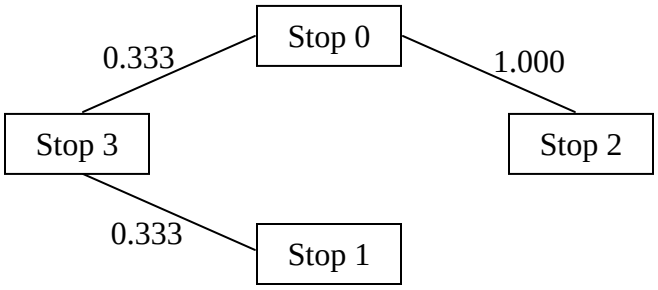
- The **handleQueries** function may invoke other additional functions that you may write to answer user queries about bus services.
- User will enter integer values for stop numbers.

Task 3

Write a function **createOptimalBusSystem** to modify the existing bus system such that all stops that are visited by buses remain visited but the total cost for the bus system is minimal. Stops that are visited by busses according to the input file data will remain unvisited in the modified bus system.

The cost of going from stop *i* to stop *j* is defined as the reciprocal of the number of buses that travels from stop *i* to stop *j*, where stops *i* and *j* are linked directly, i.e. without intermediate stops. For example, bus services 1, 3, 4, 6, and 7 travel from stop 5 to stop 6. Consequently, the cost of stop 5 to stop 6 is 1/5 or 0.2.

The total cost of a bus system is defined as the sum of the costs of all directly linked bus stops. For example, the total cost of the bus system shown below is 0.333 + 0.333 + 1.000 = 1.666.



The function **createOptimalBusSystem** is to print out the source and destination stop numbers following by the associated cost of the minimal cost system. Only source and destination pairs that form the minimal cost system need to be printed. For example, if the cost of stop 0 to stop 3 is 0.333 and the cost of stop 3 to stop 1 is 0.333, then it is only necessary to print the costs of stops 0 to 3 and 3 to 1. It is **not** necessary to print the cost of stop 0 to 1 as 0.666.

You can make the following assumptions:

- The **createOptimalBusSystem** function may print source and destination bus stop numbers in any order. For example, all of the following outputs are acceptable:

```
From 0 to 3: cost = 0.333
From 3 to 1: cost = 0.333
From 0 to 2: cost = 1.000
```

```
From 0 to 2: cost = 1.000
From 0 to 3: cost = 0.333
From 3 to 1: cost = 0.333
```

```
From 3 to 1: cost = 0.333
From 0 to 3: cost = 0.333
From 0 to 2: cost = 1.000
```

```
From 0 to 2: cost = 1.000
From 3 to 1: cost = 0.333
From 0 to 3: cost = 0.333
```

```
From 3 to 1: cost = 0.333
From 0 to 2: cost = 1.000
From 0 to 3: cost = 0.333
```

```
From 0 to 3: cost = 0.333
From 0 to 2: cost = 1.000
From 3 to 1: cost = 0.333
```

- Within each line of output, you can print the stops in any order. For example,

  `From 0 to 3: cost = 0.333`

  is equivalent to

  `From 3 to 0: cost = 0.333`

- Given a bus system, each visited stop is connected either directly or indirectly through bus services to every other visited stop in the bus system.

- You can only modify the given bus system by removing bus services between bus stops. You are not allowed to add bus services to the bus system to reduce costs between bus stops.

The output format required is:

`From <source> to <destination>: cost = <cost>`

A sample run is shown below.

Given the following sample input file, **pe.in**:

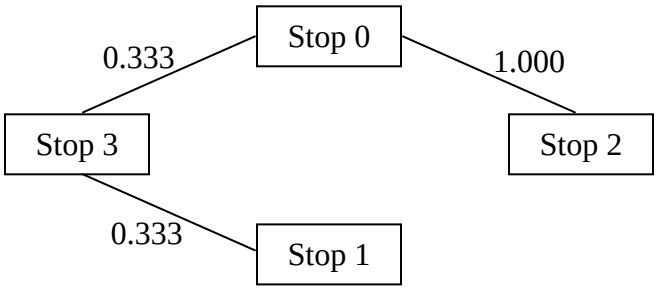
```
10
0 1 2
0 2 3
0 3 4
1 3 4
1 0 5
2 0 3
1 3 6
0 3 1
0 3 6
3 1 7
```

A sample run of your program should look like this (data entered by user are in bold):

```
   0   2   1   3   0   0   0   0
   2   0   0   3   0   0   0   0
   1   0   0   0   0   0   0   0
   3   3   0   0   0   0   0   0
   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0

Enter a destination: 100
Invalid stop number: 100
Do you wish to continue? y
Enter a destination: 5
The following buses visit destination 5:
  No buses found!
Do you wish to continue? y
Enter a destination: 0
The following buses visit destination 0:
  1  2  3  4  5  6
Do you wish to continue? n
From 0 to 3: cost = 0.333
From 3 to 1: cost = 0.333
From 0 to 2: cost = 1.000
```

The answer for Task 3 in the above example can be visualized as follows:



Your program will be tested with other test data.

Happy Programming!