Journeyman Program beta

Pre-work assignment

The Journeyman Program has been design to simulate real software development conditions whilst maximising the knowledge and experience that you acquire throughout.

Core to the philosophy of the Journeyman program lies the reality that developers mostly work in teams. Based on this premise we designed the program so that people work in teams to deliver a real and very awesome system.

This course is about your personal development and it will be what you make out of it. It is important that you work with people that will compliment your skills the best.

Considering these points we would like to ensure that attendees:

- have the required level of proficiency to get value out of a course like this,
- can organise themselves into teams of people at the same level of proficiency, and
- have the required level of commitment to see the course through

We devised a very simple programming exercise, in order most effectively determine your level of proficiency.

Do not be alarmed, it is not possible to fail the Journeyman Program.

There are 4 exercises:

- FizzBuzz
- Supermarket Pricing
- Data structures
- Train problem (Optional)

Guidance for assignment:

- Programming Languages: JavaScript, Ruby, C# or Java
- Try using different languages for different problems
- Do put the code on a public Github repo titled "nreality-journeyman-prework"
- Please make use of TDD

Keep in mind that the goal of this course is to make developers productive in their chosen platforms and not to teach the basics of these courses. Please use one of the following online courses to ramp-up:

- www.codeschool.com (Quality end-to-end courses)
- www.codecademy.com (Free end-to-end courses)
- www.pluralsight.com (Excellent individual subjects)

Journeyman Program beta
Pre-work assignment

Problem 1: FizzBuzz

(Expected to take about 30 minutes)

Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.

Problem 2: Supermarket Pricing

(Should take one evening)

http://codekata.pragprog.com/2007/01/code_kata_one_s.html

Problem 3: Data Structures

(Should take one evening)

Implemented your own list and queue from scratch. You are allowed to use the default array in the languages of choice.

- Queue
- List (Self-redimensioning array similar to the .NET List in System.Collections)

Implement a binary search on the list that you created.

Recommended reading:

- http://msdn.microsoft.com/en-us/library/hh830851(v=vs.80).aspx
- http://en.wikipedia.org/wiki/Queue_(data_structure)

Problem 4: Trains

(Should take one maybe two evenings)

The local commuter railroad services a number of towns. Because of monetary concerns, all of the tracks are 'one-way.' That is, a route from A to B does not imply the existence of a route from B to C. In fact, even if both of these routes do happen to exist, they are distinct and are not necessarily the same distance!

The purpose of this problem is to help the railroad provide its customers with information about the routes. In particular, you will compute the distance along a certain route, the number of different routes between two towns, and the shortest route between two towns. The solution must be flexible enough such that when new routes are added no code changes need to be made to the application.

Input: A directed graph where a node represents a town and an edge represents a route between two towns. The weighting of the edge represents the distance between the two towns. A given route will never appear more than once, and for a given route, the starting and ending town will not be the same town.

Output: For test input 1 through 5, if no such route exists, output 'NO SUCH ROUTE'. Otherwise, follow the route as given; do not make any extra stops! For example, the first problem means to start at city A, then travel directly to city B (a distance of 5), then directly to city C (a distance of 4).

- The distance of the route A-B-C.
- The distance of the route A-D.
- The distance of the route A-D-C.
- The distance of the route A-E-B-C-D.
- The distance of the route A-E-D.
- The number of trips starting at C and ending at C with a maximum of 3 stops. In the sample data below, there are two such trips: C-D-C (2 stops). and C-E-B-C (3 stops).
- The number of trips starting at A and ending at C with exactly 4 stops. In the sample data below, there are three such trips: A to C (via B,C,D); A to C (via D,C,D); and A to C (via D,E,B)
- The length of the shortest route (in terms of distance to travel) from A to C.
- The length of the shortest route (in terms of distance to travel) from B to B.
- The number of different routes from C to C with a distance of less than 30. In the sample data, the trips are: CDC, CEBC, CEBCDC, CDCEBC, CDEBC, CEBCEBC, CEBCEBCEBC.

Test Input:

For the test input, the towns are named using the first few letters of the alphabet from A to D. A route between two towns (A to B) with a distance of 5 is represented as AB5.

Graph: AB5, BC4, CD8, DC8, DE6, AD5, CE2, EB3, AE7

Expected Output:

- Output #1: 9
- Output #2: 5
- Output #3: 13
- Output #4: 22
- Output #5: NO SUCH ROUTE
- Output #6: 2
- Output #7: 3
- Output #8: 9
- Output #9: 9
- Output #10: ?