Berkeley Lights Programming Take Home

From the time you receive this test (from BLI), you have 7 days (168 hours) to finish. Answer a minimum of 2 problems. You are free to use any online resource; however, you must provide references.

You can address the problems using code or code with unit-tests or design diagrams or a clearly described solution (with edge cases and pitfalls) or a combination of these.

If you choose to write code, we recommend C# or Python. Please don't send us executables; send us the source code and instructions on how to compile and run it.

Problem 1. grepsharp: C# version of grep

Create a command-line utility (*grepsharp*) that will search for a text pattern in a set of text files. The return value is all the lines from the files.

Usage:

- grepsharp [OPTIONS] -f FILE1
- grepsharp [OPTIONS] -f FILE1 FILE2 FILE3 …

Options:

- -c: Return count of matching lines per file.
- -i: Prepend the output lines with filename.
- -p <PATTERN> : String pattern to search
- -r <REGEX> : Regex pattern to search (use C# regex)
- -n: Inverse the match. (Show all lines that do not match the pattern)
- -l <NUM> : Print NUM lines of output, wait for user input, then show next NUM lines.
- -C <NUM> : Print NUM lines of leading and trailing context.

Problem 2: Increment version number

Your software uses a string version number in the form “1.2.3.4”. Major version, Minor version, Build and Revision separated by a period. The indices of the versions in the string are Major:0, Minor:1, Build:2 and Revision:3. Write a method that takes as input the version number in string format and the index to increment and outputs the new version number. What are the edge cases that need to be considered? How would you test it?

Examples:

- IncrementVersion(“1.2.3.4”, 3) => “1.2.3.5”
- IncrementVersion(“1.2.3.4”, 2) => “1.2.4.0”
- IncrementVersion(“1.a.3.4”, 1) => “1.b.0.0”

Problem 3: Package Delivery Problem

You are employed at a logistics company specializing in transporting packages between businesses. Each business generates and receives packages. When a package is generated, it is tagged with its destination. For example: Business A generates a package "1:X" this is a package with ID 1 that should be delivered to Business X. Assume all packages are the same size. Your company has one small delivery truck.

Design route finding software with the following goals:

1. Section 3.1 shows a list of businesses and the distance between them in miles. Section 3.2 contains a list of packages generated at each business and their destination. Find a route, starting at the "Warehouse", to deliver all these packages. Minimize the number of visits to the same business.
2. Update the route if the truck can carry a maximum of 5 packages.
3. Update the route if the truck can carry fuel for a maximum distance of 100 miles and must return to the Warehouse for re-fueling?
4. For faster delivery you can buy a larger truck or a second truck. What information do you need to make this decision? Assume the cost of a large delivery truck is 25% less than two smaller delivery trucks.

Section 3.1: The format of all the business and distances between them is described as a comma-separated list of values. For Problem 3 a sample input is given below:

From, To:DistanceInMiles, To:DistanceInMiles, ….
Warehouse, A:50, B:15, Hub:20
Hub, C:5, D:20, F:10, G:30
A, Warehouse:50
B, Warehouse:15
C, Hub:5
D, Hub:20
E, F:10
F, E:10, G:20, Hub:10
G, F:20, Hub:30

Section 3.2: The list of packages generated is also a comma-separated list of values. Sample input is below:

Source, PackageId:Destination, PackageId:Destination
A, 1:B, 2:F, 3:Warehouse
B, 4:Warehouse, 5:Warehouse, 6:Warehouse, 7:Warehouse
Warehouse, 8:A, 9:D, 10:B, 11:C, 12:D, 15:E, 16:E, 17:G, 18:G
D, 19:A, 20:Warehouse, 21:F, 22:F

Problem 4: Order Processing

Write a program for the following scenario. Unit tests are expected.

Your code needs to process an order placed by a user. An order is always for a single product.

1. Check inventory of the product from a persistent store. We are still in the process of designing the data store. Feel free to use whatever works for now.
2. If there are enough quantities in the persistent store,
    a. Call a 3rd party payment service to verify the credit card info. For simplicity assume incoming Order contains user's credit card. Payment gateway has the following web service interface. Every time we call this interface they charge us a fee! *Boolean ChargePayment(string creditCardNumber, decimal amount).*
    b. Send an e-mail to shipping department instructing them to ship the order. Do not send real emails, while running your tests!!!