Alex Lee Developer Technical Screening

This is a technical screening to set a bar for candidates. We will use the results of these questions, your resume, and any additional information you send us to determine the next steps in the interview process.

Feel free to answer these in a separate file or document and send that to us or send it as a series of individual files, whatever makes the most sense to you. For the unit tests and other pieces (MVC/React) we would like to see how you name your methods, classes, components and how you organize your code to get a good sense of your thought process. Additionally, please provide comments on any pieces of code you think are not quite so intuitive.

1. (C#) Include Unit Tests

Write a function that accepts two strings and produces an interleaved result composed of alternating letters from each source string. Example,

Input string 1 - "abc" and string 2 "123", the output should be "a1b2c3".

2. (C#) Include Unit Tests

Find if the given string is a palindrome or not?

The user will input a string and program should print “Palindrome” or “Not Palindrome” based on whether the input string is a palindrome or not.

input: madam, output: Palindrome

input: step on no pets, output: Palindrome

input: book, output: Not Palindrome

if we pass an integer as a string parameter then also the program should give the correct output

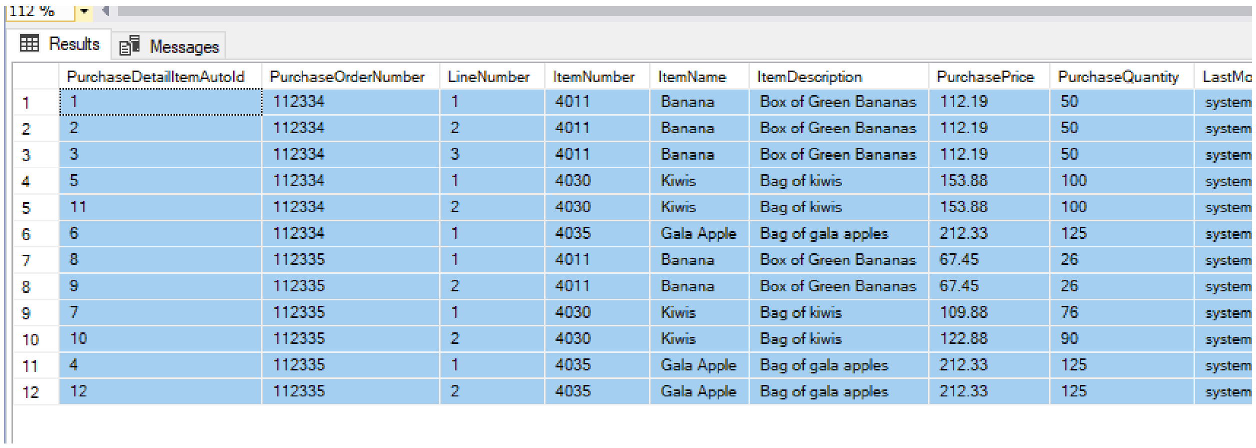
input: 1221, output: Palindrome

3. (C#)

Write a function that takes a source directory path, a search string, and a destination filename. The function should then open all the files in the given directory in parallel (using **parallelization** technique of your choice), find lines that have the search text in them, and extract and output all lines with that search text in a file with the given destination filename. At the end of execution, the function should output the number of files it processed, the number of lines it found the search text in, and the number of occurrences of that search term (don't assume once per line).

For the SQL questions please find the SQL script attached labelled, “SQLExerciseScript.sql” for creating a simple table on any database you choose.

4. (SQL) – The table PurchaseDetailItem has records that were inconsistently inserted for two different Purchase Order numbers, can you write a quick query to create a line number column per item per purchase order number? Provide the query below, Expected results pictured below:



	PurchaseDetailItemAutoId	PurchaseOrderNumber	LineNumber	ItemNumber	ItemName	ItemDescription	PurchasePrice	PurchaseQuantity	LastMo
1	1	112334	1	4011	Banana	Box of Green Bananas	112.19	50	system
2	2	112334	2	4011	Banana	Box of Green Bananas	112.19	50	system
3	3	112334	3	4011	Banana	Box of Green Bananas	112.19	50	system
4	5	112334	1	4030	Kiwis	Bag of kiwis	153.88	100	system
5	11	112334	2	4030	Kiwis	Bag of kiwis	153.88	100	system
6	6	112334	1	4035	Gala Apple	Bag of gala apples	212.33	125	system
7	8	112335	1	4011	Banana	Box of Green Bananas	67.45	26	system
8	9	112335	2	4011	Banana	Box of Green Bananas	67.45	26	system
9	7	112335	1	4030	Kiwis	Bag of kiwis	109.88	76	system
10	10	112335	2	4030	Kiwis	Bag of kiwis	122.88	90	system
11	4	112335	1	4035	Gala Apple	Bag of gala apples	212.33	125	system
12	12	112335	2	4035	Gala Apple	Bag of gala apples	212.33	125	system

5. (SQL) – The PurchaseDetailItem table also seems to have duplicate records, can you write a query to identify the duplicates inserted that have the same purchase order number, item number, purchase price and quantity that are the same? Provide the query below:

6. (SQL) – Create a stored procedure that lists out all the purchase detail records with the line number field from question #4.

(React/NextJS preferred)

(Extra points for styling the components nicely)

7. (React) – Project List Purchase Detail Results
Create a React project that lists the results of purchase details in a grid format, can just be static array list in the app itself or wire up a simple API to provide purchase detail records using the stored procedure or an Entity Framework context to get records.

Implement individual filters to change grid results by purchase order #, item #, item name and description. Implement a clear button to clear the filters and the grid results.

8. (React) – Create/Update Purchase Detail Records

(Extra points if you create a modal window to handle add/update functionality from the grid results.)

Create a page/view to add/update purchase detail records, can just modify the static array list in the app or again have API methods to do the create/update work.

(Required – but if you do React you can omit)

9. (MVC) – MVC Project List Results (Can use a local SQLExpress database and/or leave out your connection string to whatever database you use when sending the exercise back).

Create an MVC project that uses the stored procedure you wrote in question #6 to get purchase details or you can create an Entity Framework context to get records from the table. (Can use a local SQLExpress database and/or leave out your connection string to whatever database you use when sending the exercise back).

Create a view to list out the purchase detail records in a grid format. Implement individual filters to change grid results by purchase order #, item #, item name and description. Implement a clear button to result the filters and the grid results.

10. (MVC) – Insert/Update Record

(Extra points if you create a modal window to handle add/update functionality from the grid results.)

Building upon the MVC project you created in question #9 create a view and a method to add/update a purchase detail record to the PurchaseDetailItem table.