Scalable Data Infrastructures - Code Exercise 01

Overview

Being able to extrapolate the details of a problem is an important skill as a programmer. It is rare that someone will give you a list of exactly what they need your code to do. More likely, that person will give you a sketch of what they want, and it will be up to you to extract the details in a problem analysis. The first part of this code exercise will allow you to work on these problem analysis skills. In the second part of the exercise, you'll take your problem analysis from the first part and create a simple C# program that will output text to the console.

Part 1

Instructions

Below is a description of a problem. In essence, it is a word problem. Read through it and then extrapolate the details of the problem as a bulleted list. This does not mean simply putting all the sentences into separate bullets. You need to take a close look at what the problem is requiring you to do and list out the steps required to complete the process.
Your final result should be a PDF file that contains your name, the class and term, and the code exercise number along with the bulleted list that breaks down the problem into smaller bits. Compress the PDF into a zip file and upload it to this activity.

The Problem

Break down the following problem into a bulleted list:

My SDI instructor is obviously a bit nosy. He wants me to tell him a little about myself by creating a C# program that will tell him my first and last name, my degree program, and my reasons for wanting to become a developer. And I'm supposed to tell him all this in my code's output.

Things to Consider

As will be mentioned throughout this class, attention to detail is vital in programming. So, we'll start with attention to detail in exercises like this. In other words, you will lose points if your bulleted list is not in PDF format, and you will lose points if you upload the PDF instead of the zip file.

Part 2

Instructions

Using Visual Studio, start a new project, and create a new C# project/solution that will output information to the console. This output should include all of the information outlined in your problem analysis. So, any information about what the code should do can be extrapolated from the first part of this exercise.

Remember that all output should be meaningful to a user. If you simply output a list with no context, a user will not know what you are trying to say. Be sure to include enough text to give the output meaning.

The name of the project should be in the following format: LastName_FirstName_CE01.
Find the project folder on your computer and compress the entire folder into a zip file.

Create a zip file of this folder by right-clicking or CTRL+clicking the folder and selecting Compress in the pop-up menu. Make sure the name of the zip file matches the convention noted above and upload this to FSO.

Things to Consider

Look to your problem analysis to verify that you've included all the outputs required by the problem. If you ask your instructor or a lab specialist for assistance on the code, he/she is going to ask to see your problem analysis first. If you didn't do that, he/she is going to tell you to complete that task before looking at your code. So, make sure you're doing the exercise in the correct order.

Rubric: Code Exercise 1

Minimum Project Requirements

These requirements must be satisfied before any points are awarded. Failing to meet these requirements will result in a zero (0) grade.

1. The submission must be in the proper format as defined in the FSO activity.
2. The submission includes the problem analysis and the required code files. Submissions with only the Program.cs file or only the .sln file or only the .csproj file will receive a zero (0) grade.
3. You will lose 5 points if the project does not follow the naming convention described in the activity's documentation.

Topic	%	Excellent (100%)	Acceptable (80%)	Good (50%)	Fair (25%)	Poor (0%)
Problem Analysis						
Bulleted List	**10**	Problem is broken down into a bulleted list and each point represents an aspect of the problem in a short, to-the-point statement.	Problem is broken down into a bulleted list, but list items are not restated in student's own words. The bulleted points are simply copied and pasted from the problem.	Problem is broken down into a list, but list is missing bullet points.	Problem is broken down into a list, but missing more than one important bullet point or more than one point listed in a single bullet item.	Problem is not broken down into a list, bulleted or otherwise.
Problem Points Listed	**20**	All problem points are extracted from the problem and listed in the bulleted list.	Missing one important problem points in the bulleted list.	Missing two important problem points in the bulleted list.	Missing more than two problem points.	The problem points are not included in the bulleted list.
PDF Submitted	**10**	File was converted to a PDF, compressed to a Zip file, and submitted.	File was not converted to a PDF, but was compressed to a Zip file.	File was converted to a PDF file, but it was not compressed to a Zip file.		File submitted was not a PDF and was not compressed.
Coding						
Project Created	**30**	Project was correctly created, and all required project files were included in the submission.	Project was created, but some files missing from the submission.	Project was created, but some aspect of the file structure is incorrect.		Project was not created correctly.
Code Works	**30**	Project code runs and all required outputs are present.	Project code runs, but one or two outputs are missing.	Project code contains minor syntax errors but is easily fixed.	Project code contains more major syntax errors but are easily fixed.	Project code does not run.