PROG 1700 - ASSIGNMENT 1

CONSOLE APPLICATIONS, VARIABLES, INPUT AND OUTPUT

Assignment Value: 10% of overall course mark.

Due Date: **See due date designated on the Assignment 1 dropbox on Brightspace.**

Late submissions will receive the standard late submission penalty as stated in the course outline. (5% overall deduction per day late, until 60%, and 0% after assignment handed back to the class.)

Assignment Instructions:

Use Visual Studio Code to create console applications (.js files) in which you'll code the answer for each of the following problems. **You must create a new .js file for each question in this assignment.**

Submissions:

You will submit your work for this assignment via GitHub. While you will have frequent commits/pushes of your assignment code to GitHub as your work on it, the instructor needs to know which version to mark and when it was committed. So, when you have completed all assignment work, put a "Ready for Marking" comment on the last code committed into GitHub. Then submit a simple text document to the Brightspace Dropbox that contains the git Commit ID string (e.g. "b180b37") that identifies that commit. It is this Dropbox submission that will be used to determine late penalties, so make sure to do so prior to the assignment deadline.

Once you have committed the code, make sure to visit the repository page on GitHub's website to verify that the final version has been pushed to GitHub as that is where the instructor will go to get your code.

Evaluation:

To insure the greatest chance of success on this assignment, be sure to check the marking rubric contained at the end of this document or in Brightspace. The rubric contains the criteria your instructor will be assessing when marking your assignment.

Program 1 – Hipster's Local Vinyl Records

Hipster's Local Vinyl Records sell and hand-deliver vinyl records to their customers. Delivery is charged at a rate of $15 per kilometer.

All purchases are subject to a 14% sales tax.

Because this store loves retro technology, you have been asked to develop a console application solution that will enable the company's retail staff to calculate receipts. Begin by designing your solution to this problem in pseudocode, which will be submitted along with the program. The program user must be able to input the customer's name, the number of kilometers distance, and the cost of any records purchased. Once the input is provided, the program will display the customer's name and three calculated costs: delivery cost, records cost (plus tax) and total cost, as shown below.

Examples and Testing

In the section below you will be presented with at least one screenshot of a successful execution of a sample solution to the program, which should help demonstrate how your input/output on the program should work. In addition to the sample values used in the screenshot(s), additional testing values are given in a chart along with the output values that they should produce. You can expect your instructor to grade your assignment by using all of these listed input values at the very least, but keep in mind that additional values may also be used as well. **In other words, you should thoroughly test your code before submitting!**

Sample Output - Make sure your program can output data *exactly* as shown below.

```
Hipster's Local Vinyl Records - Customer Order Details

Enter the customer's name: Joe Cool
Enter the distance in kilometers for delivery: 4.5
Enter the cost of cost of records purchased: 129.95

Purchase summary for Joe Cool
Delivery Cost: $67.50
Purchase Cost: $148.14
Total Cost   : $215.64
```

Other testing values;

Distance (km)	Records Cost	Delivery Cost	Purchase Cost	Total Cost
12	$259.99	$180.00	$296.39	$476.39
7.5	$129.95	$112.50	$148.14	$260.64

Program 2 – Weekly Loan Calculator

Develop a short term loan calculator program as a console application with the following specifications. Begin by designing your solution to this problem in pseudocode, which will be submitted along with the program.

If A dollars are borrowed at r% interest compounded weekly to purchase something with weekly payments for n years, then the weekly payment is given by the formula:

If:

$$i = \frac{r}{5200}$$

Then calculate the weekly payment as:

$$\text{weekly payment} = \frac{i}{1 - (1 + i)^{-52n}} * A$$

Write a console application that calculates the weekly payment after the user gives the amount of the loan, interest rate, and number of years.

Examples and Testing

In the section below you will be presented with at least one screenshot of a successful execution of a sample solution to the program, which should help demonstrate how your input/output on the program should work. In addition to the sample values used in the screenshot(s), additional testing values are given in a chart along with the output values that they should produce. You can expect your instructor to grade your assignment by using all of these listed input values at the very least, but keep in mind that additional values may also be used as well. **In other words, you should thoroughly test your code before submitting!**

SAMPLE OUTPUT - Make sure your program can output data *exactly* as shown below.

```
Weekly Loan Calculator

Enter the amount of loan: 23500
Enter the interest rate (%): 2.8
Enter the number of years: 4

Your weekly payment will be:  $119.46
```

OTHER TESTING VALUES:

Loan Amount	Interest Rate	Number of Years	Monthly Payment
36000	3.4	6	$127.59
23500	2.8	4	$119.46

Program 3 – Imperial to Metric Conversion

Write a console program that converts a weight given in tons (imperial), stones, pounds, and ounces to the metric equivalent in metric tons, kilograms, and grams. Begin by designing your solution to this problem in pseudocode, which will be submitted along with the program.

After the numbers of tons, stones, pounds, and ounces are input by the user, the weight should be converted entirely into ounces (the lowest common denominator) and then divided by 35.274 to obtain the value in kilos. The JavaScript *parseInt* function should be used to break the total number of kilos into a whole number of metric tons and kilos. The number of grams should be displayed to one decimal place.

Required formulas:

total ounces = 35840 * tons + 224 * stone + 16 * pounds + ounces

total kilos = total ounces / 35.274

metric tons = Int(kilos/1000)

Examples and Testing

In the section below you will be presented with at least one screenshot of a successful execution of a sample solution to the program, which should help demonstrate how your input/output on the program should work. In addition to the sample values used in the screenshot(s), additional testing values are given in a chart along with the output values that they should produce. You can expect your instructor to grade your assignment by using all of these listed input values at the very least, but keep in mind that additional values may also be used as well. **In other words, you should thoroughly test your code before submitting!**

SAMPLE OUTPUT - Make sure your program can output data *exactly* as shown below.

```
Imperial To Metric Conversion

Enter the number of tons: 5
Enter the number of stone: 20
Enter the number of pounds: 2
Enter the number of ounces: 4

The metric weight is 5 metric tons, 208 kilos, and 255.4 grams.
```

OTHER TESTING VALUES:

Tons	Stone	Pounds	Ounces	Output
8	340	1	4	The metric weight is 10 metric tons, 288 kilos, and 30.8 grams.
24	78	0	10	The metric weight is 24 metric tons, 880 kilos, and 705.3 grams.

Program 1 – Hipster Record Shop

Criteria	Insufficient (0 pts)	Needs Development (1-2 pts)	Sufficient (3-4 pts)	Excellent (5 pts)	Marks	X
Pseudocode	Little to no effort was made, or contains too many errors / omissions.	A reasonable effort was made, but there are multiple areas for improvement.	A good effort was made, but at least one error or omission exists.	A comprehensive effort was made to plan out the program using pseudocode		
Input / Output	Little to no effort was made, or contains too many errors / omissions. Little to no effort was made, or contains too many errors / omissions.	A reasonable effort was made, but there are multiple areas for improvement. A reasonable effort was made, but there are multiple areas for improvement.	A good effort was made, but at least one error or omission exists. A good effort was made, but at least one error or omission exists.	All of: • The three user input values can be successfully captured through descriptive prompts • The four output lines are well-formatted and contain all expected information • Output amounts displayed using proper currency formatting (e.g. preceded by a $ symbol, two decimal places)		2
Variables & Data Types	Little to no effort was made, or contains too many errors / omissions.	A reasonable effort was made, but there are multiple areas for improvement.	A good effort was made, but at least one error or omission exists.	Solution displays strong understanding of variable usage. All data is stored using correct data types and cast to other data types when required.		
Customer Name	Little to no effort was made, or contains too many errors / omissions.	A reasonable effort was made, but there are multiple areas for improvement.	A good effort was made, but at least one error or omission exists.	Customer name is output to the console as expected		
Delivery Cost	Little to no effort was made, or contains too many errors / omissions.	A reasonable effort was made, but there are multiple areas for improvement.	A good effort was made, but at least one error or omission exists.	Delivery cost amount is correctly calculated based on variable values. Any input produces the correct output amount.		
Purchase Cost	Little to no effort was made, or contains too many errors / omissions.	A reasonable effort was made, but there are multiple areas for improvement.	A good effort was made, but at least one error or omission exists.	Purchase cost amount is correctly calculated based on variable values. Any input produces the correct output amount.		
Total Cost	Little to no effort was made, or contains too many errors / omissions.	A reasonable effort was made, but there are multiple areas for improvement.	A good effort was made, but at least one error or omission exists.	Total cost amount is correctly calculated and output to the console. Any input produces the correct output amount.		
Comments & Best Coding Practices (At least 60% of the functional requirements must be complete)	Little to no effort was made, or contains too many errors / omissions.	A reasonable effort was made, but there are multiple areas for improvement.	A good effort was made, but at least one error or omission exists.	Organizational or explanatory comments are used extensively, most are meaningful and easily understood. A consistent naming convention was used for most of the program and deviated very little. Source code was clean, consistently well-formatted and easy to read		

Total:		/45

Program 2 - Weekly Loan Calculator

Criteria	Insufficient (0 pts)	Needs Development (1-2 pts)	Sufficient (3-4 pts)	Excellent (5 pts)	Marks	X
Pseudocode	Little to no effort was made, or contains too many errors / omissions.	A reasonable effort was made, but there are multiple areas for improvement.	A good effort was made, but at least one error or omission exists.	A comprehensive effort was made to plan out the program using pseudocode		
Input / Output	Little to no effort was made, or contains too many errors / omissions.	A reasonable effort was made, but there are multiple areas for improvement. A reasonable effort was made, but there are multiple areas for improvement.	A good effort was made, but at least one error or omission exists. A good effort was made, but at least one error or omission exists.	All of: • The three user input values can be successfully captured through descriptive prompts • The single output line is well-formatted and contains all expected information • Output amount displays using proper currency formatting (e.g. preceded by a $ symbol, two decimal places)		2
Variables & Data Types	Little to no effort was made, or contains too many errors / omissions.	Solution displays little to no understanding of variable usage. Data rarely used or stored in correct data type.	Solution displays some understanding of variable usage. Most data is stored using correct data types or some errors in casting exist	Solution displays strong understanding of variable usage. All data is stored using correct data types and cast to other data types when required.		
Weekly Payment	Little to no effort was made, or contains too many errors / omissions.	A reasonable effort was made, but there are multiple areas for improvement.	A good effort was made, but at least one error or omission exists.	Weekly payment amount is correctly calculated based on variable values.		3
Comments & Best Coding Practices (At least 60% of the functional requirements must be complete)	Little to no effort was made, or contains too many errors / omissions.	A reasonable effort was made, but there are multiple areas for improvement.	A good effort was made, but at least one error or omission exists.	Organizational or explanatory comments are used extensively, most are meaningful and easily understood A consistent naming convention was used for most of the program and deviated very little Source code was clean, consistently well-formatted and easy to read		
				Total:		/40

Program 3 - Imperial to Metric Conversion						
Criteria	**Insufficient (0 pts)**	**Needs Development (1-2 pts)**	**Sufficient (3-4 pts)**	**Excellent (2 pts)**	**Marks**	**X**
Pseudocode	Little to no effort was made, or contains too many errors / omissions.	A reasonable effort was made, but there are multiple areas for improvement.	A good effort was made, but at least one error or omission exists.	A comprehensive effort was made to plan out the program using pseudocode		
Input / Output	Little to no effort was made, or contains too many errors / omissions.	A reasonable effort was made, but there are multiple areas for improvement.	A good effort was made, but at least one error or omission exists.	Both of: • The four user input values can be successfully captured through descriptive prompts • The single output line contains all expected information and formatted as shown		2
Variables & Data Types	Little to no effort was made, or contains too many errors / omissions.	A reasonable effort was made, but there are multiple areas for improvement.	A good effort was made, but at least one error or omission exists.	Solution displays strong understanding of variable usage. All data is stored using correct data types and cast to other data types when required.		
Conversion to Metric	Little to no effort was made, or contains too many errors / omissions.	A reasonable effort was made, but there are multiple areas for improvement.	A good effort was made, but at least one error or omission exists.	Conversion to metric is correctly calculated based on variable values.		3
Comments & Best Coding Practices (At least 60% of the functional requirements must be complete)	Little to no effort was made, or contains too many errors / omissions.	A reasonable effort was made, but there are multiple areas for improvement.	A good effort was made, but at least one error or omission exists.	Organizational or explanatory comments are used extensively, most are meaningful and easily understood A consistent naming convention was used for most of the program and deviated very little Source code was clean, consistently well-formatted and easy to read		
				Total:		**/40**

Assignment 1 Total: _____ / 125