CS 161A: Programming and Problem Solving I

Assignment A02 Sample Algorithmic Design Document

Make a copy before you begin (File -> Make a copy). Add the Assignment # above and complete the sections below BEFORE you begin to code. The sections will expand as you type. When you are finished, download this document as a PDF (File -> Download -> PDF) and submit to D2L.

This document contains an interactive checklist. To mark an item as complete, click on the box (the entire list will be highlighted), then right click (the clicked box will only be highlighted), and choose the checkmark.

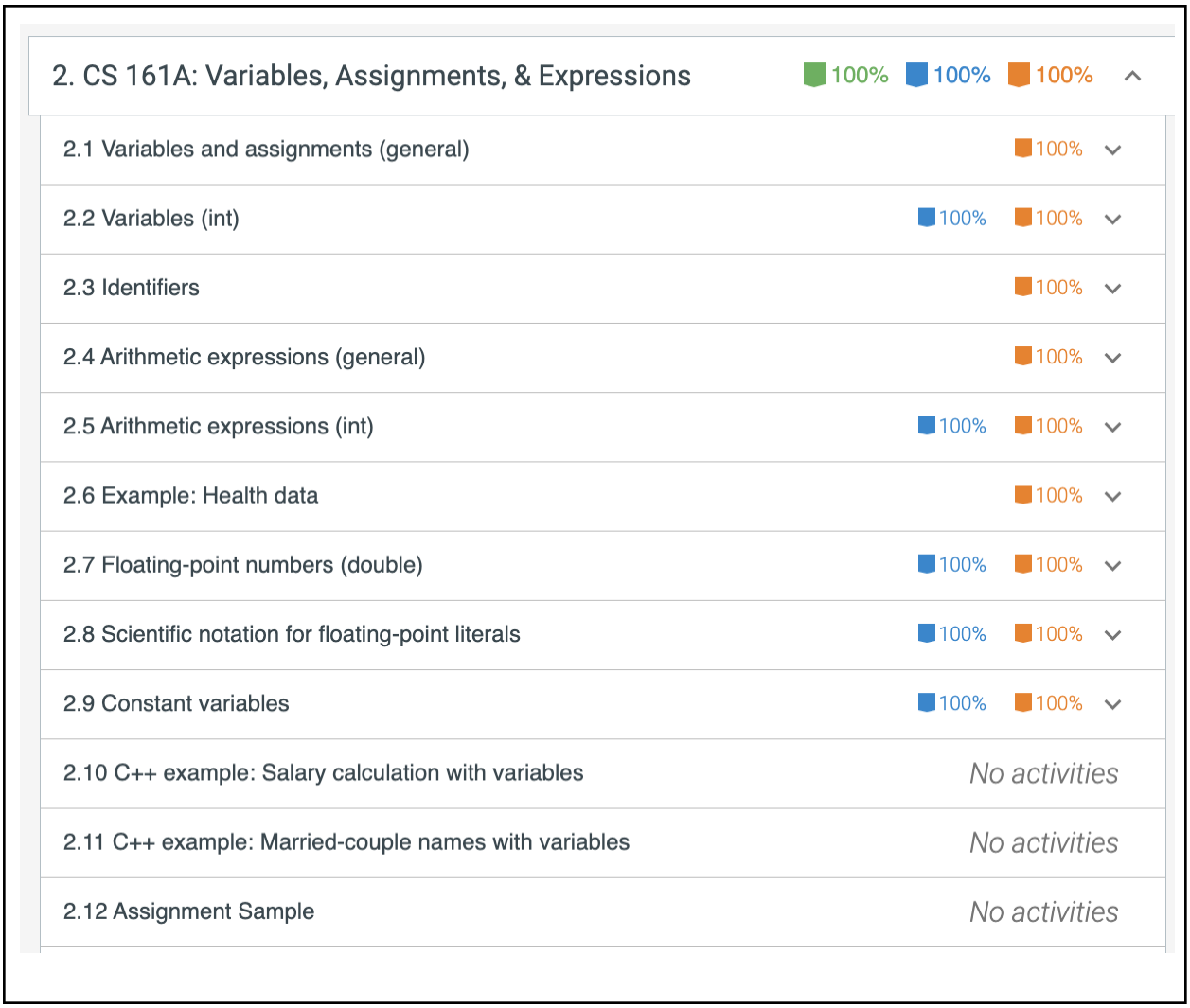
Planning your program before you start coding is part of the development process. In this document you will:

- ☑ ~~Paste a screenshot of your zyBooks Challenge and Participation %~~
- ☑ ~~Paste a screenshot of your assigned zyLabs completion~~
- ☑ ~~Write a detailed description of your program, at least two complete sentences~~
- ☑ ~~If applicable, design a sample run with test input and output~~
- ☑ ~~Identify the program inputs and their data types~~
- ☑ ~~Identify the program outputs and their data types~~
- ☑ ~~Identify any calculations or formulas needed~~
- ☑ ~~Write the algorithmic steps as pseudocode or a flowchart~~
- ☐ Tools for flowchart - Draw.io - Diagrams.net

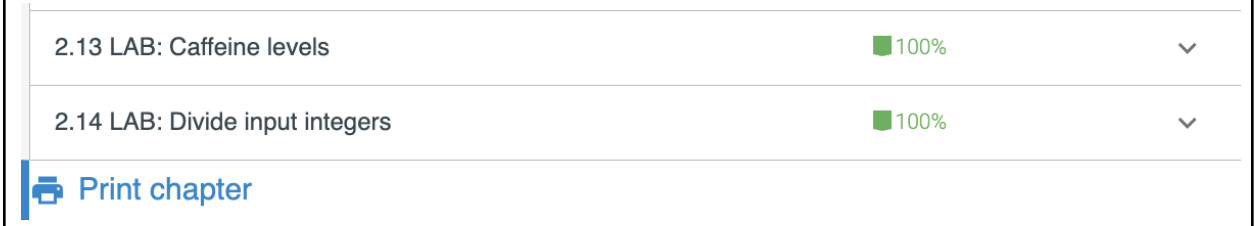
1. zyBooks

Add your zyBooks screenshots for the % and assigned zyLabs completions below. Required percentages: all **assigned** zyLabs, Challenge Activity with at least 70%, and Participation Activity with at least 80%.

Challenge and Participation % screenshot: See below



Assigned zyLabs completion screenshot:



2. Program Description

In the box below, describe the purpose of the program. You must include a detailed description with at least two complete sentences.

Program description:
This program will take a user's input of their hourly pay and weekly hours, and make the necessary deductions for them to let them know the user know their net pay, and what the deductions are.

3. Sample Run

If you are designing your own program, you will start with a sample run. Imagine a user is running your program - what will they see? What inputs do you expect, and what will be the outputs from the given inputs? Choose test data you will use to test your program. Calculate and show the expected outputs. Use the sample run to test your program.

Sample run:

```
Welcome to Gina's Weekly Payroll program…!

Please enter your employee number (numbers only): $12345
Please enter number of hours worked (whole numbers): 40
Please enter the hourly rate: $20

Your Payroll Summary:
Total Gross pay: $800.00
FICA Deductions: $61.20
Federal Tax Withholding: $120.00
Total Deductions: $181.20
Net Pay: $618.80
Thank you for using Gina's Weekly Payroll program!!
```

4. Algorithmic Design

Before you begin coding, **you must first plan out the logic** and think about what data you will use to test your program for correctness. All programmers plan before coding - this saves a lot of time and frustration! Use the steps below to identify the inputs and outputs, calculations, and steps needed to solve the problem.

Algorithmic design:
a. Identify and list all of the user input and their data types.

- **Employee id as integer:** [employeeId] The id of the employee at their job.
- **Hours Worked as integer:** [hoursWorked] The hours worked per week in whole number format.
- **Hourly Rate as a double:** [hourlyRate] The amount of $ paid per hour to the user in currency format.

b. Identify and list all of the user output and their data types.

- **Federal Withholding Rate as a double:** [federalWithholdingRate] The amount of $ the Federal Government withholds from the user's hourly pay (per week). *{in percentage format}*
- **Total Gross Pay as a double:** [totalGrossPay] The amount of money the user gets paid before any deductions. *{in currency format}*
- **FICA deduction amount as a double**: [ficaDeduction]: The amount after the rate is calculated that is actually deducted from the user's gross pay. *{in currency format}*
- **Federal Tax WIthholding Amount as a double**: [federalTaxWithholding] The amount actually deducted by the Federal taxes after the rate is calculated with user gross pay. *{in currency format}*
- **Total Deductions as a double**: [totalDeductions] The total amount of all deductions in this program from the user's gross pay. *{in currency format}*
- **FICA Deduction rate as a constant double:** [FICADEDUCT] The amount that FICA deducted from the user's pay, which is at 7.65% for Social Security and Medicare taxes. *{in percentage format}*
- **Net Pay as a double:** [netPay] What the user gets paid after all of the deductions, aka take-home pay. *{in currency format}*

c. What calculations do you need to do to transform inputs into outputs? List all formulas needed, if applicable. If there are no calculations needed, state there are no calculations for this algorithm.

- **totalGrossPay** = hoursWorked * hourlyRate;
- **ficaDeduction** = totalGrossPay * FICADEDUCT / 100;
- **federalTaxWithholding** = totalGrossPay * federalWithholdingRate;
- **totalDeductions** = ficaDeduction + federalTaxWithholding;
- **netPay** = totalGrossPay - totalDeductions;

d. Design the logic of your program using pseudocode or flowcharts. Here is where you would use conditionals, loops or functions (if applicable) and list the steps in transforming inputs into outputs. Walk through your logic steps with the test data from the assignment document or the sample run above.

1. **DISPLAY** text "Welcome to Gina's Weekly Payroll program…"

2. **DECLARE** variables **employeeId**, **hoursWorked**, **hourlyRate**, **federalWithholdingRate**, **totalGrossPay**, **ficaDeduction**, **federalTaxWithholding**, **totalDeductions**, **netPay**, and **FICADEDUCT** = 7.65
3. **DECLARE** variables as integers, doubles, and a const double for **FICADEDUCT**
4. **DISPLAY** prompt - "Please enter your employee number (numbers only):"
5. **INPUT** into **employeeId**
6. **DISPLAY** prompt - "Please enter the number of hours worked (whole numbers):"
7. **INPUT** into **hoursWorked**
8. **DISPLAY** prompt - "Please enter the hourly rate: $"
9. **INPUT** into **hourlyRate**
10. **SET** **totalGrossPay** = hoursWorked * hourlyRate;
11. **SET** **ficaDeduction** = totalGrossPay * FICADEDUCT / 100;
12. **SET** **federalTaxWitholding** = totalGrossPay * federalWithholdingRate;
13. **SET** **totalDeductions** = ficaDeduction + federalTaxWithholding;
14. **SET** **netPay** = totalGrossPay - totalDeductions;
15. **SET** fixed << setprecision(2) → To make all doubles have a 2-decimal-place limit
16. **DISPLAY** → "Your Payroll Summary:" → text only
17. **DISPLAY** → "Total Gross Pay: $" → **totalGrossPay** value
18. **DISPLAY** → "FICA Deductions: $" → **ficaDeduction** value
19. **DISPLAY** → "Federal Tax Withholding: $" → **federalTaxWithholding** value
20. **DISPLAY** → "Total deductions: $" → **totalDeductions** value
21. **DISPLAY** → "Net Pay: $" → **netPay** value
22. **DISPLAY** → "Thank you for using my weekly Payroll program!!" → text only

5. Pseudocode Syntax

Think about each step in your algorithm as an action and use the verbs below:

To do this:	**Use this verb:**	**Example:**
Create a variable	DECLARE	`DECLARE integer num_dogs`
Print to the console window	DISPLAY	`DISPLAY "Hello!"`
Read input from the user into a variable	INPUT	`INPUT num_dogs`
Update the contents of a variable	SET	`SET num_dogs = num_dogs + 1`
Conditionals		
Use a single alternative conditional	IF *condition* THEN *statement*	`IF num_dogs > 10 THEN` `DISPLAY "That is a lot of`

	statement END IF	dogs!” END IF

Use a dual alternative conditional	IF *condition* THEN
	statement
	statement
	ELSE
	statement
	statement
	END IF
		IF num_dogs > 10 THEN
		DISPLAY “You have more than 10 dogs!”
		ELSE
		DISPLAY “You have ten or fewer dogs!”
		END IF

Use a switch/case statement	SELECT *variable or expression* CASE *value_1:* *statement* *statement* CASE *value_2:* *statement* *statement* CASE *value_2:* *statement* *statement* DEFAULT: *statement* *statement* END SELECT	SELECT num_dogs CASE 0: DISPLAY “No dogs!” CASE 1: DISPLAY “One dog..” CASE 2: DISPLAY “Two dogs..” CASE 3: DISPLAY “Three dogs..” DEFAULT: DISPLAY “Lots of dogs!” END SELECT
Loops		
Loop while a condition is true - the loop body will execute 0 or more times.	WHILE *condition* *statement* *statement* END WHILE	SET num_dogs = 1 WHILE num_dogs < 10 DISPLAY num_dogs, “ dogs!” SET num_dogs = num_dogs + 1 END WHILE
Loop while a condition is true - the loop body will execute 1 or more times.	DO *statement* *statement* WHILE *condition*	SET num_dogs = 1 DO DISPLAY num_dogs, “ dogs!” SET num_dogs = num_dogs + 1 WHILE num_dogs < 10
Loop a specific number of times.	FOR *counter* = *start* TO *end* *statement* *statement* END FOR	FOR count = 1 TO 10 DISPLAY num_dogs, “ dogs!” END FOR
Functions		
Create a function	FUNCTION *return_type name (parameters)* *statement* *statement* END FUNCTION	FUNCTION Integer add(Integer num1, Integer num2) DECLARE Integer sum SET sum = num1 + num2 RETURN sum END FUNCTION

Call a function	CALL *function_name*	`CALL add(2, 3)`
Return data from a function	RETURN *value*	`RETURN 2 + 3`