

CS 161A: Programming and Problem Solving I

Assignment A05 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](https://draw.io) - [Diagrams.net](https://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:

12. CS 161B: File Input/Output	 100%	 100%	 100%	^
12.1 File input	 100%	 100%	v	
12.2 File output	 100%	 100%	v	

Assigned zyLabs completion screenshot:

12.7 LAB: All-consuming (File Input)

100%



12.8 LAB: Cereal Analysis

100%



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 reads data from a file containing job titles, employment types, salaries, and remote work ratios. It analyzes the data to identify the position with the highest salary and provides a summary of the total number of positions and the average salary.

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 Data Analysis Program!
This program reads data from a file, performs analysis, and
displays results.

Position with the highest salary:
Job Title: Research Engineer
Position: FT
Salary: $275000
Remote Ratio: 0%

Summary Analysis:
Total Positions: 10
Average Salary: $149664.00
```

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:
<ul style="list-style-type: none">Identify and list all of the user input and their data types.<ul style="list-style-type: none">File items.txt (contains information about various job positions, including job titles, employment types (FT or CT), salaries, and remote work ratios)
<ul style="list-style-type: none">Identify and list all of the user output and their data types.<ul style="list-style-type: none">maxSalaryIndex (integer): stores the index of the job position with the highest salary (in the for loop)totalPositions (integer): stores the total number of positions listed in the data file items.txtaverageSalary (double): stores the avg salary calculated from all positions listed in the data file items.txtFUNCTIONS:<ol style="list-style-type: none">analyzeData (void): function responsible for identifying the job position with the highest salary and printing out the detailssumAnalysis (void): function that calculates total number of positions and average salary and prints out summary
<ul style="list-style-type: none">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.<ul style="list-style-type: none">maxSalaryIndex: if (salaries[i] > salaries[maxSalaryIndex]) then maxSalaryIndex = iaverageSalary = totalSalary / count
<ul style="list-style-type: none">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.
// Define constants

```

DECLARE ITEMS AS 10
DECLARE MAXCHAR AS 51

// Function prototypes
FUNCTION welcome()
FUNCTION openFile(inFile)
FUNCTION loadData(inFile, items[], positions[], salaries[], ratios[])
FUNCTION analyzeData(items[], positions[], salaries[], ratios[], count)
FUNCTION sumAnalysis(items[], positions[], salaries[], ratios[], count)

// Main function
FUNCTION main()
    DECLARE inFile AS ifstream
    DECLARE count AS Integer
    DECLARE items[ITEMS][MAXCHAR]
    DECLARE positions[ITEMS][MAXCHAR]
    DECLARE salaries[ITEMS]
    DECLARE ratios[ITEMS]

    // Display welcome message
    welcome()

    // Open file and check if it opens
    IF NOT openFile(inFile) THEN
        OUTPUT "File did not open! Program terminating!!"
        EXIT
    END IF

    // Load data from file
    count = loadData(inFile, items, positions, salaries, ratios)
    CLOSE inFile

    // Analyze the data
    analyzeData(items, positions, salaries, ratios, count)
    sumAnalysis(items, positions, salaries, ratios, count)

END FUNCTION

// Function to display welcome message
FUNCTION welcome()
    OUTPUT "Welcome to the Data Analysis Program!"
    OUTPUT "This program reads data from a file, performs analysis, and displays
results."
END FUNCTION

// Function to open file and check if it opens
FUNCTION openFile(inFile)
    OPEN "items.txt" AS inFile
    RETURN inFile.is_open()
END FUNCTION

```

```

// Function to load data from file into arrays
FUNCTION loadData(inFile, items[], positions[], salaries[], ratios[])
    DECLARE count AS Integer
    SET count = 0

    WHILE count < ITEMS AND NOT inFile.end_of_file()
        READ items[count], positions[count], salaries[count], ratios[count] FROM inFile
        INCREMENT count
    END WHILE

    RETURN count
END FUNCTION

// Function to analyze the data
FUNCTION analyzeData(items[], positions[], salaries[], ratios[], count)
    DECLARE maxSalaryIndex AS Integer

    // Find the position with the highest salary
    SET maxSalaryIndex = 0
    FOR i FROM 1 TO count - 1
        IF salaries[i] > salaries[maxSalaryIndex] THEN
            SET maxSalaryIndex = i
        END IF
    END FOR

    // Output the position with the highest salary
    OUTPUT "Position with the highest salary:"
    OUTPUT "Job Title: " + items[maxSalaryIndex]
    OUTPUT "Position: " + positions[maxSalaryIndex]
    OUTPUT "Salary: $" + salaries[maxSalaryIndex]
END FUNCTION

// Function to perform summary analysis
FUNCTION sumAnalysis(items[], positions[], salaries[], ratios[], count)
    DECLARE totalPositions AS Integer
    DECLARE totalSalary AS Double
    DECLARE averageSalary AS Double

    // Calculate total positions and total salary
    SET totalPositions = count
    FOR i FROM 0 TO count - 1
        INCREMENT totalSalary BY salaries[i]
    END FOR

    // Calculate average salary
    SET averageSalary = totalSalary / count

    // Output summary analysis
    OUTPUT "Summary Analysis:"

```

```

    OUTPUT "Total Positions: " + totalPositions
    OUTPUT "Average Salary: $" + AVERAGE_SALARY (formatted to two decimal
places)
END FUNCTION

```

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 <i>condition</i> THEN <i>statement</i> <i>statement</i> END IF	IF num_dogs > 10 THEN DISPLAY "That is a lot of dogs!" END IF
Use a dual alternative conditional	IF <i>condition</i> THEN <i>statement</i> <i>statement</i> ELSE <i>statement</i> <i>statement</i> 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 <i>variable or expression</i> CASE <i>value_1</i> : <i>statement</i> <i>statement</i> CASE <i>value_2</i> : <i>statement</i> <i>statement</i> CASE <i>value_2</i> : <i>statement</i> <i>statement</i> DEFAULT: <i>statement</i> <i>statement</i>	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

	END SELECT	
Loops		
Loop while a condition is true - the loop body will execute 0 or more times.	WHILE <i>condition</i> <i>statement</i> <i>statement</i> 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 <i>statement</i> <i>statement</i> WHILE <i>condition</i>	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 <i>counter</i> = <i>start</i> TO <i>end</i> <i>statement</i> <i>statement</i> END FOR	FOR count = 1 TO 10 DISPLAY num_dogs, " dogs!" END FOR
Functions		
Create a function	FUNCTION <i>return_type</i> <i>name (parameters)</i> <i>statement</i> <i>statement</i> END FUNCTION	FUNCTION Integer add(Integer num1, Integer num2) DECLARE Integer sum SET sum = num1 + num2 RETURN sum END FUNCTION
Call a function	CALL <i>function_name</i>	CALL add(2, 3)
Return data from a function	RETURN <i>value</i>	RETURN 2 + 3