CS 161A/B: Programming and Problem Solving I

# Algorithm 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.ne](https://www.draw.io/)t

## 1. Academic integrity Acknowledgement

**I understand that acts of academic dishonesty may be penalized to the full extent allowed by the Portland Community College Student Conduct Code, including receiving a failing grade for this exam. I recognize that I am responsible for understanding the provisions of the PCC Student Conduct Code as they relate to this Course Challenge Exam.**

Kyle Noyes

## 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 calculate the fare a passenger must pay when given the passengers name, number of carry on bags, and number of checked bags. |

## 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 the Complex Airfare Calculator!  Please enter your name: **Cool Person**  Hello Cool Person!  Please enter age of the passenger: **41**  How many carry-on bags do you have (0 or 1)? **1**  How many checked bags do you have? **0**  Your airfare is: $310  Thank you for flying ComplexAir!  Welcome to the Complex Airfare Calculator!  Please enter your name: **Kyle Noyes**  Hello Kyle Noyes!  Please enter age of the passenger: **26**  How many carry-on bags do you have (0 or 1)? **1**  How many checked bags do you have? **2**  Your airfare is: $335  Thank you for flying ComplexAir! |

## 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.

Use the pseudocode syntax shown in the document, supplemented with English phrases if necessary. **Do not include any implementation details (e.g. source code file names, class or struct definitions, or language syntax)**. Do not include any C++ specific syntax or data types.

|  |
| --- |
| **Algorithmic design:** |
| 1. Identify and list all of the user input and their data types. Include a variable name, data type, and description. Data types include string, integer, floating point, (single) character, and boolean. Data structures should be referenced by name, e.g. “array of integer” or “array of string (for CS161B and up). |
| passengerName as Str  passengerAge as Int  numBagsCarryOn as Int  numBagsChecked as Int |
| 1. Identify and list all of the user output and their data types. Include a variable name, data type, and description. Data types include string, integer, floating point, (single) character, and boolean. Data structures should be referenced by name, e.g. “array of integer” or “array of string” (for CS161B and up). |
| Simple calculations:  fareTotal as Int; it is the total calculated fare |
| 1. 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. Formulae should reference the variable names from step a and step b as applicable. |
| fareTotal = fareTotal + FARE\_ADULT;  fareTotal = fareTotal + FARE\_ELDER;  fareTotal = fareTotal + (numBagsCarryOn \* FARE\_CARRY\_ON);  Complex calculation block (grouped)  checkedBagProcessing = checkedBagProcessing - 1;  fareTotal = fareTotal + FARE\_CHECKED\_BAG\_FIRST;  checkedBagProcessing = checkedBagProcessing - 1;  fareTotal = fareTotal + (checkedBagProcessing \* FARE\_CHECKED\_BAG\_SECOND); |
| 1. 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. **Use the syntax shown at the bottom of this document and plain English phrases. Do not include any implementation details (e.g. file names) or C++ specific syntax.** |
| START  DECLARE Global Int FARE\_ADULT = 300  DECLARE Global Int FARE\_ELDER = 290  DECLARE Global Int FARE\_CARRY\_ON = 10  DECLARE Global Int FARE\_CHECKED\_BAG\_FIRST = 25  DECLARE Global Int FARE\_CHECKED\_BAG\_SECOND = 50  DECLARE String passengerName = ""  DECLARE Int passengerAge = 0  DECLARE Int numBagsCarryOn = 0  DECLARE Int numBagsChecked = 0  DECLARE Int bagPriceProcessing = 0  DECLARE Int runState = 0  DECLARE Int fareTotal = 0  // Init user  DISPLAY "Please enter your name: "  INPUT userName  DISPLAY "Hello {passengerName}!"  // Pax age  DISPLAY "Please enter age of the passenger: "  INPUT passengerAge  IF passengerAge < 0 || passengerAge > 100  - - SET runState = 1  - - DISPLAY "Age {passengerAge} is invalid! Sorry you cannot fly ComplexAir!"  // Carry-on Bags  IF runState = 0  - - How many carry-on bags do you have (0 or 1)?  - - INPUT numBagsCarryOn  - - IF numBagsCarryOn > 1  - - - - SET runState = 1  - - - - DISPLAY "You have too many carry-on bags! Sorry you cannot fly ComplexAir!"  - - ELSE IF numBagsCarryOn < 0  - - - - SET runState = 1  - - - - DISPLAY "Number of carry-on bags is invalid! Sorry you cannot fly ComplexAir!"  // Checked Bags  IF runState = 0  - - DISPLAY "How many checked bags do you have?"  - - INPUT numBagsChecked  - - SET bagPriceProcessing = numBagsChecked  - - IF numBagsChecked < 0  - - - - SET runState = 1  - - - - DISPLAY "Number of checked bags is invalid! Sorry you cannot fly ComplexAir!"  - - ELSE IF numBagsChecked > 5  - - - - SET runState = 1  - - - - DISPLAY "Too many checked bags (5 max)! Sorry you cannot fly ComplexAir!"  // Fare Calculation  IF runState = 0  - - IF passengerAge > 2 AND passengerAge < 60  - - - - SET fareTotal = fareTotal + FARE\_ADULT  - - IF passengerAge >= 60  - - - - SET fareTotal = fareTotal + FARE\_ELDER  - -  - - SET totalFare = totalFare + (numBagsCarryOn \* FARE\_CARRY\_ON)  - -  - - IF numBagsChecked > 1  - - - - bagPriceProcessing = bagPriceProcessing - 1  - - - - SET totalFare = totalFare + FARE\_CHECKED\_BAG\_FIRST  - - - - bagPriceProcessing - 1  - - - - SET totalFare = totalFare + (bagPriceProcessing \* FARE\_CHECKED\_BAG\_SECOND)  - -  - - DISPLAY "Your airfare is: {totalFare}"  - - DISPLAY "Thank you for flying ComplexAir!"  END |

## 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*  *statement*  END IF | IF num\_dogs > 10 THEN  DISPLAY “That is a lot of 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 = star*t 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 |