# **Assignment xx 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 and submit with your Assignment to D2L (File -> Download -> PDF). The sections will expand as you type.

## 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%.

|  |
| --- |
| **zyLabs, Challenge, and Participation % Screenshot:** |
|  |

|  |
| --- |
| **Assigned zyLabs completion Screenshot:** |
| N/A |

## Assignment

|  |
| --- |
| **Program description:** |
| This program will prompt you to enter a list of numbers with the first entry being the total amount of numbers to check. Once all numbers are inputted, this program will calculate the percentage of all numbers that are even vs. odd and display the result |

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:** |
| 1. Identify all of the user input. What are the data types of the inputs? Define the input variables. |
| In function getPercentEven:  No user inputs are required  In function main  Int, arrSize // allows creation of dynamic list  Int, numArr all in one line separated by spaces // a foor loop adds each item into array |
| 1. Describe the program output. What is displayed to the user? What are the data types of the output? Define the output variables. |
| The program will output a string advising the user that what percentage of the lsit contains even numbers. The final part of the string contains the formatted calculated percent. |
| 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. |
| In function getPercentEven:  calcOddEven = numList[i] % 2  percEven = (numEven / indexTotal) \* 100  In function main  No calculations are processed |
| 1. Design the logic of your program using pseudocode or flowcharts. See pseudocode syntax at the bottom of this document. Here is where you would use conditionals, loops, functions or array constructs (if applicable) and list the steps in transforming inputs into outputs. Walk through your logic steps with the test data from the assignment document. |
| START  FUNCTION Double getPercentEven(Int indexTotal, Int \*numList)  - - DECLARE Int index  - - DECLARE Double numEven = 0  - - DECLARE Dobule percEven  - - DECALRE Int calcOddEven  - -  - - FOR index = 0 TO index = indexTotal  - - - - calcOddEven = numList[index]  - - - -  - - - - IF calcOddEven % 2 == 0  - - - - - - numEven = numEven + 1  - -  - - SET percEven = (numEven / indextotal) \* 100  - -  - - return percEven  FUNCTION Int Main(void)  - - DECLARE Int arrSize  - - DECLARE Int i  - - DECLARE Double percentEven  - -  - - DISPLAY "Welcome to my Final."  - - DISPLAY "How many numbers would you like in your list?"  - -  - - INPUT arrSize  - -  - - DECLARE Array numArr[arrSize]  - -  - - IF arrSize > 0  - - - - DISPLAY "Please enter the values for your list separated by a space: "  - - - -  - - - - FOR i = 0 TO i = arrSize  - - - - - - INPUT numArr[i]  - - - -  - - - - SET percentEven = CALL Function getPErcentEven(arrSize, numArr)  - -  - - ELSE  - - - - percentEven = 0  - -  - - DISPLAY "The percent of even values: %.1f%%", percentEven  - - DISPLAY "Thank you for using my program!"  - -  - - RETURN 0  END FUNCTION  END |
| 1. Include 2 Sample Program Runs for your program using your own set of data. This data set must be different from my Sample Runs in the Assignment document. This process is similar to Unit Testing and will help you test your program better. |
| **Sample Program Run 1:**    **Sample Program Run 2:** |

## 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 |