# **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 take a low number, a high number, and a multiples number to get the amount of times that multiple can fit in the range of the program. |

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. |
| Int userLow; // The low number  Int userHigh; // The high number  Int userMult; // the value to check multiples of  Char userChoice; // Char to check the Y/N continue question against |
| 1. Describe the program output. What is displayed to the user? What are the data types of the output? Define the output variables. |
| All data out will be strings. We have the following blocks of output given based on the conditions.  Error conditions:  “Invalid entry, the {low/high/multi} number {first/second/third} must be less than the {compare}”  “Invalid entry, the {low/high/multi} number {first/second/third} can not be a negative number.”  “Please try again”  Success condition:  “The number of multiples of userMult between userLow and userHigh is: {numDivCount}” |
| 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. |
| numDivCount = (numDiff % userMult); // Get remainders |
| 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  DECLARE Int userLow;  DECLARE Int userHigh;  DECLARE Int userMult;  DECLARE Int numDiff;  DECLARE Int numDivCount;  DECLARE Str userChoice[256] = "0";  DECLARE keepGoing;  DISPLAY "Hello! Welcome to my midterm for CS-133U-11053"  DISPLAY "Developer: Kyle Noyes"  DISPLAY "Date: February 18th, 2024"  DISPLAY "Welcome to this multiples-counter."  WHILE userChar != "n"  - - keepGoing = 0  - - DISPLAY "Please enter 3-whole numbers with the first being your low number, second being"  - - DISPLAY "the high number, and third being the multiples to check between low and high"  - -  - - DISPLAY "Input: "  - - INPUT userLow, userHigh, userMult  - -  - - // Validate data inputs  - - IF userLow >= userHigh  - - - - DISPLAY "Invalid entry, the low number (first) must be less than the high"  - - - - DISPLAY "number. Please try again."  - - - - keepGoing = 1  - -  - - IF userMult >= userHigh  - - - - DISPLAY "Invalid entry, the multiples (third) must be less than the high"  - - - - DISPLAY "number. Please try again."  - - - - keepGoing = 1  - -  - - IF userMult >= userHigh  - - - - DISPLAY "Invalid entry, the multiples (third) can not be zero."  - - - - DISPLAY "number. Please try again."  - - - - keepGoing = 1  - -  - - IF userLow < 0  - - - - DISPLAY "Invalid entry, the low number (first) can not be a negative"  - - - - DISPLAY "number. Please try again."  - - - - keepGoing = 1  - -  - - IF userHigh < 0  - - - - DISPLAY "Invalid entry, the high number (second) can not be a negative"  - - - - DISPLAY "number. Please try again."  - - - - keepGoing = 1  - -  - - IF userMult < 0  - - - - DISPLAY "Invalid entry, the multiples number (third) can not be a negative"  - - - - DISPLAY "number. Please try again."  - - - - keepGoing = 1  - -  - - IF keepGoing == 0:  - - - - SET numDiff = userHigh - userLow  - - - - SET numDivCount = numDiff / userMult  - - - - DISPLAY "The number of multiples of {userMulti} between {userLow} and {userHigh} is: {numDivCount}"  - -  - - ELSE:  - - - - PASS  - -  - - DISPLAY "Would you like to continue? ("Y/N")"  - - INPUT userChoice  DISPLAY "Thank you for using my multiples check program!"  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 |