CS 161A: Programming and Problem Solving I

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. 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

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:
❏ Write a program to output the winners of a Rock Collecting Competition. Prompt the user for three contestants: input their names as strings and the number of rocks collected as integers.

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 Rock Collector Championships!

Enter player 1 name: Gordan Freeman
How many rocks did Gordan Freeman collect? -9
Invalid amount. 0 will be entered.

Enter player 2 name: Link
How many rocks did Link collect? 45

Enter player 3 name: D. Va
How many rocks did D. Va collect? 45

Link and D. Va are tied for first place.
Gordan Freeman is in second place!

The average number of rocks collected by the top
three players is 30.00 rocks!

Congratulations Rock Collectors!
```

```
Welcome to the Rock Collector Championships!

Enter player 1 name: King Dedede
How many rocks did King Dedede collect? 57

Enter player 2 name: Samus
How many rocks did Samus collect? 102

Enter player 3 name: Kirby
How many rocks did Kirby collect? 62

Samus is in first place!
Kirby is in second place.
King Dedede is in third place.

The average number of rocks collected by the top
three players is 73.67 rocks!

Congratulations Rock Collectors!
```

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.

Player1 strings, Player2 strings, Player3 strings Rock1 int Rock 2 int Rock3 int
b. Identify and list all of the user output and their data types.
First place, second place, third place, avg numbe, tied, welcome message and goodbye message.
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.
Avg = (rock1+rock2+rock3)/3
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.

Declare player1,2,3 strings

Declare rock1,2,3 int

Print Welcome to the Lock Collector Championships!

Print “Enter player 1 name:” input player1 string

Print “How many rocks did “ << player1 string’ << “ collect ?” Input rock 1

Print “Enter player 2 name:” input player2 string

Print “How many rocks did “ << player2 string’ << “ collect ?” Input rock 2

Print “Enter player 3 name:” input player3 string

Print “How many rocks did “ << player3 string’ << “ collect ?” Input rock 3

If rock1 equal rock 2 and rock 1 equal rock 3 then print “it is a three way tie!!”

If else rock 1 > rock 2 and rock1 > 3

Print ‘player1’ is find first place

If else rock1 > rock 2 and rock1 < 3

Print player 3 is in second place.

Print player 2 is in third place.

Else rock1 < rock 2 and rock1 rock 1 >3

Repeat for all the possible outcomes using if else if statements.

Print thank you for using this program.

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* = *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`