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.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:
N/A

Assigned zyLabs completion screenshot:
N/A

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 is to output the winners of a Rock Collecting Competition. The program will prompt the user for 3 contestant names (strings), and the number of the rocks they collected (integers). There will be warnings for things such as rock amounts that are less than 0. Then, the program will determine the first, second, and third place winners. It will account for three way and two way ties. There will also be a section that calculates the average number of rocks collected (double with two decimal places).

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 Rock Collector Championship!

Enter player 1 name: Gordon 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

First place: It’s a tie between Link and D. Va!

Second place: Gordon Freeman

Third place: N/A (due to tie)
```

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

Congratulations Rock Collectors!

——————————————————————————————————————————————————————————————————————————————————————————

Welcome to Gina’s Rock Collector Championship!

Enter player 1 name: Steve Buscemi

How many rocks did Steve Buscemi collect? 99

Enter player 2 name: Yoshi

How many rocks did Yoshi collect? 98

Enter player 3 name: Odo

How many rocks did Odo collect? 100

First place: Odo

Second place: Steve Buscemi

Third place: Yoshi

The average number of rocks collected by the top three players is 99.00
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.

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:
a. 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).

playerOne (string literal): first contestant entry in the program, identified by their name

playerTwo (string literal): second contestant entry in the program, identified by their name

playerThree (string literal): third contestant entry in the program, identified by their name

playerOneRocks (integer): first contestant number of rocks found for the contest

playerTwoRocks (integer): second contestant number of rocks found for the contest

playerThreeRocks (integer): third contestant number of rocks found for the contest

b. 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).

firstPlace (string literal): contestant with the most rocks, and each player that ties

secondPlace (string literal): contestant with the 2nd most rocks, and each player that ties

thirdPlace (string literal): contestant with the 3rd most rocks, or N/A

avgRocks (double): average amount of rocks between the 3 contestant entries

NUM_PLAYERS (constant integer): this is the set number of players at 3

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. Formulae should reference the variable names from step a and step b as applicable.

Many conditionals listed in section D

avgRocks = ((playerOneRocks + playerTwoRocks + playerThreeRocks/NUM_PLAYERS)

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.

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.

```
DISPLAY welcome message
DISPLAY Enter player 1 name: message
INPUT playerOne (use getline)
DISPLAY rocks of playerOne message
  IF playerOne < 0 THEN
      DISPLAY Invalid error message
      SET playerOneRocks = 0
  END IF
(use cin.ignore() here)
DISPLAY Enter player 2 name: message
INPUT playerTwo (use getline)
DISPLAY rocks of playerTwo message
   IF playerTwoRocks < 0 THEN
       DISPLAY Invalid error message
       SET playerTwoRocks = 0
  END IF
(use cin.ignore() here)
DISPLAY Enter player 3 name: message
INPUT playerThree (use getline)
DISPLAY rocks of playerThree message
    IF playerThreeRocks < 0 THEN
        DISPLAY invalid error message
        SET playerTwoRocks = 0
    END IF
(use cin.ignore() here)
```

```
IF playerOneRocks > playerTwoRocks && playerOneRocks > playerThreeRocks THEN
    SET firstPlace = playerOne
    IF playerTwoRocks > playerThreeRocks THEN
        SET secondPlace = playerTwo
        SET thirdPlace = playerThree
    ELSE IF playerThreeRocks > playerTwoRocks
       SET secondPlace = playerThree
       SET thirdPlace = playerTwo
    ELSE
       DISPLAY secondPlace tie message between players 2 & 3
       DISPLAY thirdPlace message of N/A due to tie
    END IF (sub if)
ELSE IF playerTwoRocks > playerOneRocks && playerTwoRocks > playerThreeRocks THEN
      SET firstPlace = playerTwo
      IF playerOneRocks > playerThreeRocks THEN
          SET secondPlace = playerOne
          SET thirdPlace = playerThree
     ELSE IF playerThreeRocks > playerOneRocks
          SET secondPlace = playerThree
          SET thirdPlace = playerOne
     ELSE
          DISPLAY tie message between playerOne & playerThree
          DISPLAY thirdPlace message of N/A due to tie
      END IF (sub if)
ELSE IF playerThreeRock > playerOneRocks && playerThreeRocks > playerTwoRocks
          SET firstPlace = playerThree
          IF playerOneRocks > playerTwoRocks THEN
              SET secondPlace = playerOne
```

```
                SET thirdPlace = playerTwo
            ELSE IF playerTwoRocks > playerOneRocks
                SET secondPlace = playerTwo
                SET thirdPlace = playerOne
            ELSE
                DISPLAY tie message between playerOne and playerTwo
                DISPLAY thirdPlace display message as N/A due to tie
        END IF (sub if)
ELSE IF playerOneRocks = playerTwoRocks && playerOneRocks > playerThreeROcks
        DISPLAY firstPlace tie message between playerOne & playerTwo
        DISPLAY playerThree
        DISPLAY thirdPlace N/A message due to tie
ELSE IF playerOneRocks = playerThreeRocks && playerOneRocks > playerTwoRocks
        DISPLAY firstPlace tie message between playerOne & playerThree
        DISPLAY secondPlace = playerTwo
        DISPLAY thirdPlace N/A message due to tie
ELSE IF playerTwoRocks = playerThreeROcks && playerTwoRocks > playerOneRocks
        DISPLAY firstPlace tie message between playerTwo & playerThree
        DISPLAY secondPlace = playerOne
        DISPLAY thirdPlace N/A message due to tie
ELSE
      DISPLAY firstPlace 3 way tie message
      DISPLAY secondPlace N/A message due to tie
      DISPLAY secondPlace N/A message due to tie
END IF (parent if)


DISPLAY firstPlace message
DISPLAY secondPlace message
DISPLAY thirdPlace message
(set precision(2) here, fixed)
SET avgRocks = playerOneRocks + playerTwoRocks + playerThreeRocks / NUM_PLAYERS
DISPLAY average # of rocks message
DISPLAY congrats end message
```

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`