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.

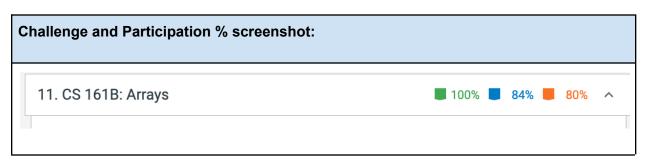
This document contains an interactive checklist. To mark an item as complete, click on the box (the entire list will be highlighted), then right click (the clicked box will only be highlighted), and choose the checkmark.

Planning your program before you start coding is part of the development process. In this document you will:

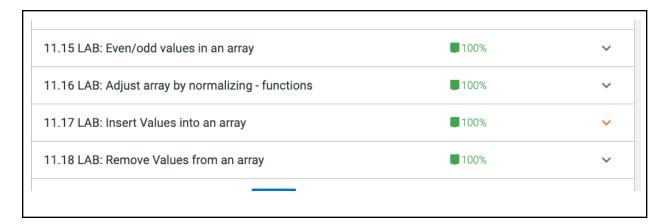
\checkmark	Paste a screenshot of your zyBooks Challenge and Participation %
\checkmark	Paste a screenshot of your assigned zyLabs completion
\checkmark	Write a detailed description of your program, at least two complete sentences
\checkmark	If applicable, design a sample run with test input and output
	Identify the program inputs and their data types
	Identify the program outputs and their data types
\checkmark	Identify any calculations or formulas needed
\checkmark	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%.



Assigned zyLabs completion screenshot:	



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 uses a user's input of their grades (on a 0-4 scale) and shows the letter grade in a list next to the score upon output. There will also be a list printed that shows the user the scores in ascending order, and another line that shows the median score.

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.

Welcome to my Parallel Arrays program! Please enter the list of scores (-1 to end input:) Valid scores are between 0 and 4 inclusive. >> 3.5 >> 2.7 >> 3.3 >> 2.5 >> 3.2 >> 1.5 >> 4.0 >> 3.7 >> -1

```
Your stats are as below:
The list of scores and their grades are:
3.5 A
2.7 в
3.3 B
2.5 C
3.2 B
1.5 D
4.0 A
3.7 A
The list sorted by scores in ascending order:
1.5 D
2.5 C
2.7 в
3.2 B
3.3 B
3.5 A
3.7 A
4.0 A
The median score is 3.25
Thank you for using my Parallel Arrays program!!
Welcome to my Parallel Arrays program!
Please enter the list of scores (-1 to end input:)
Valid scores are between 0 and 4 inclusive.
>> 3.5
>> 2.7
>> 3.3
>> 4.5
Invalid score! Please try again!!
>> abc
Invalid score! Please try again!!
>> 3.2
>> 1.5
>> 4.0
>> 3.7
>> 4.0
>> -1
Your stats are as below:
The list of scores and their grades are:
3.5 A
2.7 B
```

```
3.3 B
3.2 B
1.5 D
4.0 A
3.7 A
4.0 A
The list sorted by scores in ascending order:
1.5 D
2.7 в
3.2 B
3.3 B
3.5 A
3.7 A
4.0 A
4.0 A
The median score is 3.40
Thank you for using my Parallel Arrays program!!
```

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).

scores (double) - user can input scores in 0.0 format

scores[MAX VALUES] (double array) - array representing user scores

&count (integer) - represents # of elements in the arrays that updates as scores are read

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).

grades[MAX_VALUES] (char) - calculated by the user input for scores variable (with many conditions)

count (integer) - used to count grades and use to display lists for user and median medianScore (double) - used to store and display the median score of all scores

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.

for the median:

mid = count / 2

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.

FUNCTION void

welcome()

DISPLAY welcome message & prompt

END FUNCTION

FUNCTION void

readScores(double scores[], int &count)

END FUNCTION

FUNCTION void

readDouble (string prompt, double &num)

END FUNCTION

FUNCTION void

void calcGrade(double scores[], char grade[], int count) **END FUNCTION FUNCTION** void void printList (double scores[], char grades[], int count) **END FUNCTION FUNCTION** void sort (double scores[], char grade[], int count) **END FUNCTION FUNCTION** double median (double scores[], int count) END FUNCTION DECLARE const integer MAX_VALUES = 20 **FUNCTION** int main() DECLARE double scores[MAX VALUES] DECLARE char grades[MAX_VALUES] DECLARE integer count = 0 DECLARE double medianScore = 0.0 **CALL welcome** CALL readScores (scores, count) CALL calcGrade(scores, grades, count) DISPLAY your stats text CALL **printList**(scores, grades, count) CALL sort(scores, grades, count) CALL printList(scores, grades, count) SET double medianScore to: CALL median(scores, count) DISPLAY thank you text **END PROGRAM** // Function Definitions below **CALL void welcome** DISPLAY welcome text, enter scores prompt

```
CALL void readScores
 DECLARE double score, integer &count
 DISPLAY message to user about inputting scores
 CALL readDouble(">> ", score)
WHILE score is not -1 or MAX_VALUES (aka 20)
   SET scores array with count++ equal to score
   IF count < MAX_VALUES (aka 20)
      CALL readDouble(">> ", score)
   ELSE
     DISPLAY List is now full message
CALL void readDouble(string prompt, double &num)
  DISPLAY prompt
  INPUT num
  WHILE num is less than 0 and NOT -1 or greater than 4
      DISPLAY invalid score message
      CLEAR BUFFER
      INPUT num
      CLEAR BUFFER
CALL void calcGrade
 DECLARE int i = 0
FOR i = 0, i < count, i++
   IF scores[i] is between 3.3 and 4.0
      SET grade[i] = 'A'
   ELSE IF scores[i] is between 2.7 and 3.3
      SET grade[i] = 'B'
   ELSE IF scores[i] is between 1.9 and 2.7
      SET grade[i] = 'C'
   ELSE IF scores[i] is between 1.1 and 1.0
      SET grade[i] = 'D'
  ELSE IF scores[i] is between 0.0 and 1.1
      SET grade[i] = 'F'
  END IF
CALL void printList(double scores[], char grades[], int count)
```

```
DECLARE i = 0
  FOR i = 0, i < count, i ++
              DISPLAY scores[i] and grades [i] with a " " inbetween them
CALL void sort(double scores[], char grade[], int count)
  DECLARE minIndex (integer) = 0
  DECLARE tempScore (double) = 0.0
  DECLARE tempGrade (char) = ' '
  DECLARE i (integer) = 0
  DECLARE j (integer) = 0
  FOR i = 0, i < count, i++
     SET minIndex = i
     FOR j = i + 1, j < count, j++
       IF scores[j] < scores[minIndex]</pre>
         SET minIndex = j
       END IF
    IF minIndex does not equal i
     SET tempScore = scores[minIndex]
     SET scores[minIndex] = scores[i]
     SET scores[i] = tempScore
    SET tempGrade = grade[minIndex]
    SET grade[minIndex] = grade[i]
    SET grade[i] = tempGrade
CALL double median (double scores[], int count)
  DECLARE mid (integer) = 0
  DECLARE medianScore (double) = 0.0
  IF count modulo 2 is 1
      RETURN scores[count/2]
  ELSE
     SET mid = count / 2
     RETURN scores[mid - 1] + scores[mid] / 2.0
 DISPLAY median score message + medianScore
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	<pre>IF num_dogs > 10 THEN DISPLAY "That is a lot of dogs!" END IF</pre>			
Use a dual alternative conditional	IF condition THEN statement statement ELSE statement statement statement	<pre>IF num_dogs > 10 THEN</pre>			
Use a switch/case statement	SELECT variable or expression CASE value_1: statement statement CASE value_2: statement statement CASE value_2: statement CASE value_2: statement DEFAULT: statement	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	WHILE condition statement	SET num_dogs = 1 WHILE num_dogs < 10			

execute 0 or more times.	statement END WHILE	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			