CS 162: Computer Science II

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.

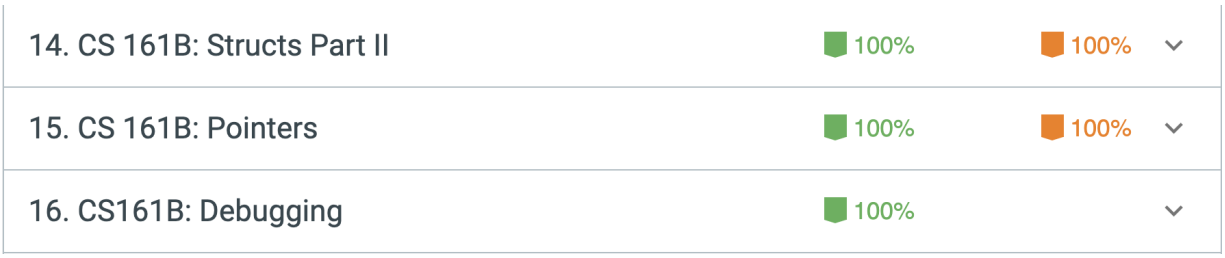
Planning your program before you start coding is part of the development process. In this document you will:

- ☑ ~~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~~

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:



Assigned zyLabs completion screenshot:

14. CS 161B: Structs Part II	100%	100%
15. CS 161B: Pointers	100%	100%
16. CS161B: Debugging	100%	

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:
Program designed to keep track of activities. Users can view the list of activities or search for activities based on name, type, or location. They can also add and remove activities from the list. The app will load the list of activities from an external .txt file and modify the .txt file based on user input.

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.
Do not simply copy the sample run from the assignment instructions!

Sample run:

```
Welcome!
This program will help you manage your activities.

Pick an option from below:

(a)Add a new activity
(b)List activities by name
(c)List activities by location
(d)List activities by Type
(e)Remove an activity
(f)Search by activity name
```

```
(q)Quit

b
```
1. Catan;Epic Gaming;Easy;4;Games

2. Oil Painting;Fun Studios;Tricky;6;Arts

3. Pottery;Fun Studios;Easy;7;Arts

4. Skiing;Mt Hood Meadows;Difficult;6;Athletics

5. Snowboarding;Mt Hood Meadows;Not for Faint of heart;8;Athletics

6. Wine Making;Umpqua Valley;Complicated;9;Food

```
Pick an option from below:

(a)Add a new activity
(b)List activities by name
(c)List activities by location
(d)List activities by Type
(e)Remove an activity
(f)Search by activity name
(q)Quit

p
Invalid option!! Please try again!

Pick an option from below:

(a)Add a new activity
(b)List activities by name
(c)List activities by location
(d)List activities by Type
(e)Remove an activity
(f)Search by activity name
(q)Quit

d
Enter Type number(0-Athletics, 1-Food, 2-Arts, 3-Games, and
4-Others): 0
```

Skiing;Mt Hood Meadows;Difficult;6;Athletics

Snowboarding;Mt Hood Meadows;Not for Faint of heart;8;Athletics

```
Pick an option from below:

(a)Add a new activity
(b)List activities by name
(c)List activities by location
(d)List activities by Type
(e)Remove an activity
(f)Search by activity name
(q)Quit

C
Enter location name: Hood Meadows

1. Skiing;Mt Hood Meadows;Difficult;6;Athletics
2. Snowboarding;Mt Hood Meadows;Not for Faint of heart;8;Athletics


Pick an option from below:

(a)Add a new activity
(b)List activities by name
(c)List activities by location
(d)List activities by Type
(e)Remove an activity
(f)Search by activity name
(q)Quit

e
1. Catan;Epic Gaming;Easy;4;Games
2. Oil Painting;Fun Studios;Tricky;6;Arts
3. Pottery;Fun Studios;Easy;7;Arts
4. Skiing;Mt Hood Meadows;Difficult;6;Athletics
5. Snowboarding;Mt Hood Meadows;Not for Faint of heart;8;Athletics
6. Wine Making;Umpqua Valley;Complicated;9;Food


Pick the index to remove: 4

Activity removed!
1. Catan;Epic Gaming;Easy;4;Games
2. Oil Painting;Fun Studios;Tricky;6;Arts
```

3. Pottery;Fun Studios;Easy;7;Arts

4. Snowboarding;Mt Hood Meadows;Not for Faint of heart;8;Athletics

5. Wine Making;Umpqua Valley;Complicated;9;Food

```
Pick an option from below:

(a)Add a new activity
(b)List activities by name
(c)List activities by location
(d)List activities by Type
(e)Remove an activity
(f)Search by activity name
(q)Quit

a
Enter the activity name (50 characters or less): Rowing
Enter the activity location (50 characters or less): Oaks Amusement
Park
Enter the activity level : Tricky
Enter the activity rating : aaa
Invalid rating! Please enter a valid rating!
Enter the activity rating : 8
Enter Type number(0-Athletics, 1-Food, 2-Arts, 3-Games, and
4-Others): 0

Activity added!
```

1. Catan;Epic Gaming;Easy;4;Games

2. Oil Painting;Fun Studios;Tricky;6;Arts

3. Pottery;Fun Studios;Easy;7;Arts

4. Rowing;Oaks Amusement Park;Tricky;8;Athletics

5. Snowboarding;Mt Hood Meadows;Not for Faint of heart;8;Athletics

6. Wine Making;Umpqua Valley;Complicated;9;Food

```
Pick an option from below:

(a)Add a new activity
(b)List activities by name
(c)List activities by location
(d)List activities by Type
(e)Remove an activity
```

```
(f)Search by activity name
(q)Quit

f
Enter the activity name (50 characters or less): Snowboarding

Activity found!
5. Snowboarding;Mt Hood Meadows;Not for Faint of heart;8;Athletics

Pick an option from below:

(a)Add a new activity
(b)List activities by name
(c)List activities by location
(d)List activities by Type
(e)Remove an activity
(f)Search by activity name
(q)Quit

f
Enter the activity name (50 characters or less): Skiing
Activity not found!!

Pick an option from below:

(a)Add a new activity
(b)List activities by name
(c)List activities by location
(d)List activities by Type
(e)Remove an activity
(f)Search by activity name
(q)Quit

q

Activities written to file! Thank you for using my 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 variables 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”.
- Char option - String name, location, level - Int rating, type
b. Identify and list all of the user output variables 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”.
- String name, location, level - Int rating, type
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.
- Addition/subtraction to add/remove activities
d. Design the logic of your program using pseudocode. 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. Do not include any implementation details (e.g. file names) or C++ specific syntax.**

ACTIVITY.H

1. DECLARE const CAP = 30
2. DECLARE const MAXCHAR = 50
3. STRUCT Activity
    a. name as a char array
    b. location as a char array
    c. level as as a char array
    d. rating as an integer
    e. type as an integer
4. Function Prototypes
    a. loadData - to load data from file into the array of structs
    b. printData - to print from the array of structs
    c. addActivity - to add new activity to activities.txt
    d. removeActivity - to remove activity from activity array
    e. searchByName - prompts user for name, returns structs with matching name
    f. searchByType - prompts user for type, returns structs with matching type
    g. searchByLocation - prompts user for location, returns structs with matching location
    h. printActivityType - maps integer to corresponding string

MAIN.H

1. Function Prototypes
    a. readInt - determines input is a positive integer
    b. readString - determines input does not contain a semicolon
    c. isLessThan - helper function utilized by loadData for sorting
    d. displayMenu - displays options to user
    e. readOption - reads user input
    f. executeCommand - calls function associated with user input
    g. writeData - writes data to external file

ACTIVITY.CPP

5. FUNCTION loadData (filestream, activityList[])
    a. DECLARE count
    b. WHILE not end of file
        i. INPUT name
        ii. INPUT location
        iii. INPUT level
        iv. INPUT rating
        v. INPUT type
    c. END WHILE LOOP

END FUNCTION loadData()

6. FUNCTION printData(activityList[], count)
    a. DISPLAY column headers
    b. FOR LOOP from i = 0 till count
        i. DISPLAY name
        ii. DISPLAY location
        iii. DISPLAY level
        iv. DISPLAY rating
        v. DISPLAY type
    c. END FOR LOOP

    END FUNCTION printData

7. FUNCTION addActivity
    a. DISPLAY prompt
    b. INPUT activityList[count].name
    c. DISPLAY prompt
    d. INPUT activityList[count].location
    e. DISPLAY prompt
    f. INPUT activityList[count].level
    g. DISPLAY prompt
    h. INPUT activityList[count].rating
    i. DISPLAY prompt
    j. INPUT activityList[count].type
    k. SET Count++;

    END FUNCTION

8. FUNCTION removeActivity
    a. IF (count == 0)
        i. DISPLAY You don't have any items in your cart, please add an activity”
        ii. RETURN
    b. DECLARE delIndex
    c. DISPLAY "Which activity would you like to remove?”
    d. INPUT delIndex;
    e. SET delIndex -= 1;
    f. WHILE (delIndex < 1 || delIndex > count)
        i. DISPLAY "Invalid item. Please select a valid item to remove."
        ii. DISPLAY "Which item would you like to remove? "
        iii. INPUT delIndex
        iv. SET delIndex -= 1
    g. FOR (int i = delIndex; i < count - 1; i++)
        i. SET activityList[i] = activityList[i + 1]
        ii. SET count–

END FUNCTION

9. FUNCTION searchByName
    a. DEFINE searchName[MAXCHAR];
    b. DEFINE count
    c. DISPLAY "Please enter activity name"
    d. INPUT searchName
    e. FOR from i = 0 till size
        i. IF(location[i] == searchName)
            1. DISPLAY activity[i]
        ii. ELSE
            1. Count++
    f. END FOR LOOP
    g. IF(count == size)
        i. DISPLAY “Activity Not Found!”

END FUNCTION

10. FUNCTION searchByLocation
    a. DEFINE searchLocation[MAXCHAR];
    b. DEFINE count
    c. DISPLAY "Please enter activity location"
    d. INPUT searchLocation
    e. FOR from i = 0 till size
        i. IF(location[i] == searchLocation)
            1. DISPLAY activity[i]
        ii. ELSE
            1. Count++
    f. END FOR LOOP
    g. IF(count == size)
        i. DISPLAY “Activity Not Found!”

END FUNCTION searchByLocation

11. FUNCTION searchByType
    a. DEFINE searchType[MAXCHAR];
    b. DEFINE count
    c. DISPLAY "Please enter activity type"
    d. INPUT searchType
    e. FOR from i = 0 till size
        i. IF(type[i] == searchType)
            1. DISPLAY activity[i]
        ii. ELSE
            1. Count++
    f. END FOR LOOP

g. IF(count == size)
    i. DISPLAY “Activity Not Found!”

END FUNCTION searchByType

12. FUNCTION printActivityType
    a. SELECT tempNumber
        i. CASE 0
            1. activityListType = “Athletics”
        ii. CASE 1
            1. activityListType = “Food”
        iii. CASE 2
            1. activityListType = “Arts”
        iv. CASE 3
            1. activityListType = “Games”
        v. CASE 4
            1. activityListType = “Others”

END FUNCTION printActivityType

MAIN.CPP

13. FUNCTION readInt
    a. DISPLAY prompt
    b. INPUT temp
    c. WHILE (!cin || temp < 0)
        i. DISPLAY "Invalid number! Please try again!!!"
        ii. DISPLAY prompt
        iii. INPUT temp
    d. RETURN temp

END FUNCTION readInt

14. FUNCTION readString
    a. DISPLAY prompt;
    b. INPUT tempString
    c. FOR i = 0 to tempString.length
        i. IF(temp[i] == ';')
            1. DISPLAY "Invalid string! Please remove all commas from entry!"
            2. CALL readString
    d. END FOR LOOP

END FUNCTION readString

15. FUNCTION isLessThan
    a. IF (strcmp(activityName1, activityName2) < 0) {
        i. RETURN true
    b. ELSE
        i. RETURN false

END FUNCTION isLessThan

16. FUNCTION displayMenu
    a. DISPLAY "Pick an option from below:"
    b. DISPLAY "(a)Add a new activity"
    c. DISPLAY "(b)List activities by name"
    d. DISPLAY "(c)List activities by location"
    e. DISPLAY "(d)List activities by Type"
    f. DISPLAY "(e)Remove an activity”
    g. DISPLAY "(f)Search by activity name"
    h. DISPLAY "(q)Quit"

END FUNCTION displayMenu

17. FUNCTION readOption() {
    a. DECLARE input;
    b. INPUT input;
    c. RETURN input;

END FUNCTION readOption()

18. FUNCTION executeCommand
    a. SELECT tempNumber
        i. CASE a
            1. CALL addActivity()
            2. CALL loadData()
            3. CALL printData()
            4. BREAK
        ii. CASE b
            1. CALL loadData()
            2. CALL printData()
            3. BREAK
        iii. CASE c
            1. CALL searchByLocation()
            2. BREAK
        iv. CASE d
            1. CALL searchByType()
            2. BREAK
        v. CASE e

1. CALL printData()
2. CALL removeActivity()
3. CALL printData()
4. BREAK

vi. CASE f
1. CALL searchByName()
2. BREAK

vii. CASE q
1. BREAK

viii. DEFAULT
1. DISPLAY “Invalid Option”

END FUNCTION executeCommand

19. FUNCTION writeData
    a. OPEN outFile
    b. FOR int i =0 until size
        i. INPUT activityList[i].name;
        ii. INPUT activityList[i].location;
        iii. INPUT activityList[i].level;
        iv. INPUT activityList[i].rating;
        v. INPUT activityList[i].type;
    c. END FOR LOOP
    d. CLOSE outFile
    e. DISPLAY "Activities written to file! Thank you for using my program!!"

    END FUNCTION writeData

20. FUNCTION main()
    a. DECLARE inFile
    b. DECLARE outFile
    c. DECLARE size
    d. DECLARE activityList[]
    e. DECLARE option
    f. DISPLAY “Welcome”
    g. CALL loadData()
    h. DO
        i. CALL displayMenu()
        ii. SET option = readOption()
        iii. CALL executeCommand()
    i. WHILE (option != q)
    j. CALL writeData()

    END FUNCTION main()

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	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`