

# 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:

17. CS 162 Classes Part I	 100%	 100%	 100%	▼
18. CS 162 Classes Part II	 100%	 100%	 100%	▼

#### Assigned zyLabs completion screenshot:

17. CS 162 Classes Part I	 100%	 100%	 100%	▼
18. CS 162 Classes Part II	 100%	 100%	 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.

<b>Algorithmic design:</b>
<ul style="list-style-type: none"> <li>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".</li> </ul>
<ul style="list-style-type: none"> <li>Char option</li> <li>String name, location, level</li> <li>Int rating, type</li> </ul>
<ul style="list-style-type: none"> <li>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".</li> </ul>
<ul style="list-style-type: none"> <li>String name, location, level</li> <li>Int rating, type</li> </ul>
<ul style="list-style-type: none"> <li>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.</li> </ul>
<ul style="list-style-type: none"> <li>Addition/subtraction to add/remove activities</li> </ul>
<ul style="list-style-type: none"> <li>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.</li> </ul> <p><b>Use the syntax shown at the bottom of this document. Do not include any implementation details (e.g. file names) or C++ specific syntax.</b></p>
<b>ACTIVITY.H</b>
1. DECLARE const MAXCHAR = 50

## 2. CLASS Activity

### a. PRIVATE

- i. activityName as a char array
- ii. activityLocation as a char array
- iii. activityLevel as as a char array
- iv. activityRating as an integer
- v. activityType as an integer

### b. PUBLIC

- i. Constructors
  - 1. Activity - default constructor
  - 2. Activity - constructor w/ params
- ii. Mutators
  - 1. setActivityName - assign char array to activityName
  - 2. setActivityLocation - assign char array to activityName
  - 3. setActivityLevel - assign char array to activityName
  - 4. setActivityRating - assign char array to activityName
  - 5. setActivityType - assign char array to activityName
- iii. Accessors
  - 1. getActivityName - assign char array to activityName
  - 2. getActivityLocation - assign char array to activityName
  - 3. getActivityLevel - assign char array to activityName
  - 4. getActivityRating - assign char array to activityName
  - 5. getActivityType - assign char array to activityName
- iv. printActivity - Display single activity to console
- v. writeActivity - Write single activity to outFile

## ACTIVITYLIST.H

### 3. DECLARE const CAP = 30

### 4. CLASS ActivityList

#### a. PRIVATE

- i. activityList as Activity array
- ii. size as int

#### b. PUBLIC

- i. Constructors
  - 1. ActivityList - default constructor
  - 2. ActivityList - constructor w/ params
- ii. Destructors
  - 1. ActivityList
- iii. Database Functions
  - 1.

#### c. name as a char array

### 5. Function Prototypes

- a. addActivity - Add activity to ActivityList



- b. removeActivity - Remove activity from ActivityList
- c. searchByName - Prompts user for name, returns activities with matching name
- d. searchByType - Prompts user for type, returns activities with matching type
- e. searchByLocation - Prompts user for location, returns activities with matching location
- f. showList - Displays list of activities
- g. writeData - Writes data to external file

## 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
- h. welcomeMessage - displays welcome message
- i. goodbyeMessage - displays goodbye message
- j. addActivity - displays prompts and calls setter methods
- k. retrieveActivityType - maps type number to type string

## ACTIVITY.CPP

### 6. DEFAULT CONSTRUCTOR Activity

- a. SET activityName = "unknown"
- b. SET activityLocation = "unknown"
- c. SET activityLevel = "unknown"
- d. SET activityRating = "unknown"
- e. SET activityType = "unknown"

END CONSTRUCTOR

### 7. CONSTRUCTOR Activity

- a. SET activityName = newActivityName
- b. SET activityLocation = newActivityLocation
- c. SET activityLevel = newActivityLevel
- d. SET activityRating = newActivityRating
- e. SET activityType = newActivityType

END CONSTRUCTOR

## MUTATOR FUNCTIONS

### 8. FUNCTION setActivityName

- a. activityName = newActivityName

END FUNCTION setName

9. FUNCTION setLocation

a. setName = newLocation

END FUNCTION setLocation

10. FUNCTION setLevel

a. setLocation = newLevel

END FUNCTION setLevel

11. FUNCTION setRating

a. setRating = newRating

END FUNCTION setRating

12. FUNCTION setType

a. setType = newType

END FUNCTION setType

#### ACCESSOR FUNCTIONS

13. FUNCTION getName

a. returnName = setName

END FUNCTION getName

14. FUNCTION getLocation

a. returnLocation = setLocation

END FUNCTION getLocation

15. FUNCTION getLevel

a. returnName = setLevel

END FUNCTION getLevel

16. FUNCTION getRating

a. returnRating = setRating

END FUNCTION getRating

17. FUNCTION getType

a. returnType = setType

END FUNCTION getType

18. FUNCTION printActivity
- a. DISPLAY activityName
  - b. DISPLAY activityLocation
  - c. DISPLAY activityLevel
  - d. DISPLAY activityRating
  - e. DISPLAY activityType

END FUNCTION printActivity

19. FUNCTION writeActivity
- a. OUTPUT activityName
  - b. OUTPUT activityLocation
  - c. OUTPUT activityLevel
  - d. OUTPUT activityRating
  - e. OUTPUT activityType

END FUNCTION writeActivity

#### **ACTIVITYLIST.CPP**

20. DEFAULT CONSTRUCTOR ActivityList
- a. Size = 0

END CONSTRUCTOR

21. CONSTRUCTOR ActivityList
- a. SET size = 0
  - b. DECLARE inFile
  - c. DECLARE activity
  - d. DECLARE tempString
  - e. DECLARE tempNumber
  - f. WHILE(inFile)
    - i. CALL setActivityName
    - ii. CALL setActivityLocation
    - iii. CALL setActivityLevel
    - iv. CALL set ActivityRating
    - v. CALL setActivityType
  - g. END WHILE LOOP

END CONSTRUCTOR

22. DESTRUCTOR ActivityList

END DESTRUCTOR

23. FUNCTION addActivity
- a. DECLARE i = 0

- b. DECLARE tempActivityName
- c. DECLARE newActivityName
- d. CALL getActivityName
- e. FOR i = 0 until size
  - i. CALL getActivityName()
  - ii. IF tempActivityName != newActivityName
    - 1. BREAK
- f. END FOR LOOP
- g. FOR j = size until j > i
  - i. activityList[i] = activityList[j - 1]
- h. END FOR LOOP
- i. SET activityList[i] = activity
- j. SET size = size + 1
- k. DISPLAY "activity added"

END FUNCTION addActivity

#### 24. FUNCTION searchByName

- a. DECLARE searchName
- b. DECLARE tempName
- c. DECLARE count = 1
- d. DISPLAY "Please enter activity name"
- e. INPUT searchName
- f. CALL getActivityName
- g. FOR from i = 0 till size
  - i. IF(location[i] == searchName)
    - 1. DISPLAY activity[i]
  - ii. ELSE
    - 1. Count++
- h. END FOR LOOP
- i. IF(count == size)
  - i. DISPLAY "Activity Not Found!"

END FUNCTION

#### 25. FUNCTION searchByLocation

- a. DEFINE searchLocation[MAXCHAR];
- b. DEFINE count
- c. DISPLAY "Please enter activity location"
- d. INPUT searchLocation
- e. CALL getActivityLocation
- f. FOR from i = 0 till size
  - i. IF(location[i] == searchLocation)
    - 1. DISPLAY activity[i]

- ii. ELSE
    - 1. Count++
- g. END FOR LOOP
- h. IF(count == size)
  - i. DISPLAY "Activity Not Found!"

END FUNCTION searchByLocation

26. FUNCTION searchByType
- a. DEFINE searchType[MAXCHAR];
  - b. DEFINE count
  - c. DISPLAY "Please enter activity type"
  - d. INPUT searchType
  - e. CALL retrieveActivityType
  - f. FOR from i = 0 till size
    - i. IF(type[i] == searchType)
      - 1. DISPLAY activity[i]
    - ii. ELSE
      - 1. Count++
  - g. END FOR LOOP
  - h. IF(count == size)
    - i. DISPLAY "Activity Not Found!"

END FUNCTION searchByType

27. FUNCTION writeData
- a. DECLARE outFile
  - b. OPEN outFile
  - c. FOR i = 0 until size
    - i. CALL writeActivity
  - d. END FOR LOOP

END FUNCTION writeDate

28. FUNCTION showList
- a. FOR i = 0 until size
    - i. DISPLAY i
    - ii. CALL printActivity()
  - b. END FOR LOOP

END showList

29. FUNCTION removeActivity
- a. DECLARE delIndex
  - b. CALL readInt()
  - c. IF (delIndex < size)

```

        i.   FOR i = delIndex until i < size
              1.  activityList[i - 1] = activityList[i]
        ii.  END FOR LOOP
    d.  SET size = size - 1
    e.  DISPLAY "Activity Removed"

END removeActivity

```

## MAIN.CPP

```

30. 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

31. 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

32. FUNCTION isLessThan
    a.  IF (activityName1 = activityName2) {
        i.   RETURN true
    b.  ELSE
        i.   RETURN false

END FUNCTION isLessThan

33. 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

34. FUNCTION readOption() {

- a. DECLARE input;
- b. INPUT input;
- c. RETURN input;

END FUNCTION readOption()

35. FUNCTION executeCommand

- a. SELECT tempNumber
  - i. CASE a
    - 1. CALL addActivity()
    - 2. CALL activityList.addActivity()
    - 3. CALL activityList.showList()
    - 4. BREAK
  - ii. CASE b
    - 1. CALL showList()
    - 2. BREAK
  - iii. CASE c
    - 1. CALL searchByLocation()
    - 2. BREAK
  - iv. CASE d
    - 1. CALL searchByType()
    - 2. BREAK
  - v. CASE e
    - 1. CALL showList()
    - 2. CALL removeActivity()
    - 3. CALL showList()
    - 4. BREAK
  - vi. CASE f
    - 1. CALL searchByName()
    - 2. BREAK
  - vii. CASE q
    - 1. BREAK
  - viii. DEFAULT
    - 1. DISPLAY "Invalid Option"

END FUNCTION executeCommand

36. 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

37. FUNCTION welcomeMessage

- a. DISPLAY "Welcome"
- b. DISPLAY "This program will help you manage your activities"

END FUNCTION

38. FUNCTION goodbyeMessage

- a. DISPLAY "Activities written to file! Thank you for using my program!!"

END FUNCTION

39. FUNCTION retrieveActivityType

- 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

40. FUNCTION addActivity

- a. DECLARE letter
- b. DECLARE tempName, tempLocation, tempLevel, tempType
- c. DECLARE tempRating, tempIntType
- d. DISPLAY prompt
- e. INPUT tempName
- f. DISPLAY prompt



- g. INPUT tempLocation
- h. DISPLAY prompt
- i. INPUT tempLevel
- j. DISPLAY prompt
- k. INPUT tempRating
- l. DISPLAY prompt
- m. INPUT tempIntType
- n. CALL retrieveActivityType()
- o. CALL setActivityName()
- p. CALL setActivityLocation()
- q. CALL getActivityLevel()
- r. CALL setActivityRating()
- s. CALL setActivityType()

END FUNCTION addActivity

41. FUNCTION main()

- a. DECLARE fileName
- b. DECLARE option
- c. CALL welcomeMessage()
- d. CALL ActivityList
- e. DO
  - i. CALL displayMenu()
  - ii. SET option = readOption()
  - iii. CALL executeCommand()
- f. WHILE (option != q)
- g. CALL writeData()
- h. CALL goodbyeMessage()

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
<b>Conditionals</b>		
Use a single alternative conditional	IF <i>condition</i> THEN <i>statement</i> <i>statement</i> END IF	IF num_dogs > 10 THEN DISPLAY "That is a lot of dogs!" END IF
Use a dual alternative conditional	IF <i>condition</i> THEN <i>statement</i> <i>statement</i> ELSE <i>statement</i> <i>statement</i> 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 <i>variable or expression</i> CASE <i>value_1</i> : <i>statement</i> <i>statement</i> CASE <i>value_2</i> : <i>statement</i> <i>statement</i> CASE <i>value_2</i> : <i>statement</i> <i>statement</i> DEFAULT: <i>statement</i> <i>statement</i> 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
<b>Loops</b>		
Loop while a condition is true - the loop body will execute 0 or more times.	WHILE <i>condition</i> <i>statement</i> <i>statement</i> 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 <i>statement</i> <i>statement</i> WHILE <i>condition</i>	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 <i>counter</i> = <i>start</i> TO <i>end</i> <i>statement</i> <i>statement</i> END FOR	FOR count = 1 TO 10 DISPLAY num_dogs, " dogs!" END FOR
<b>Functions</b>		

Create a function	<b>FUNCTION</b> <i>return_type</i> <i>name (parameters)</i> <i>statement</i> <i>statement</i> <b>END FUNCTION</b>	<pre> FUNCTION Integer add(Integer num1, Integer num2)     DECLARE Integer sum     SET sum = num1 + num2     RETURN sum END FUNCTION </pre>
Call a function	<b>CALL</b> <i>function_name</i>	<pre>CALL add(2, 3)</pre>
Return data from a function	<b>RETURN</b> <i>value</i>	<pre>RETURN 2 + 3</pre>