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.

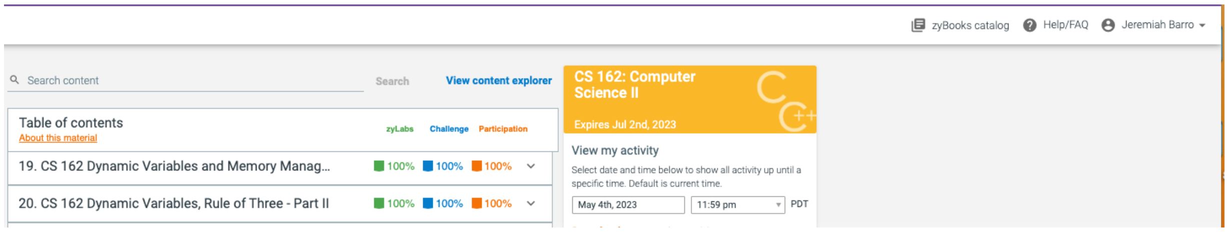
Planning your program before you start coding is part of the development process. In this document you will:

- ❏ 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:

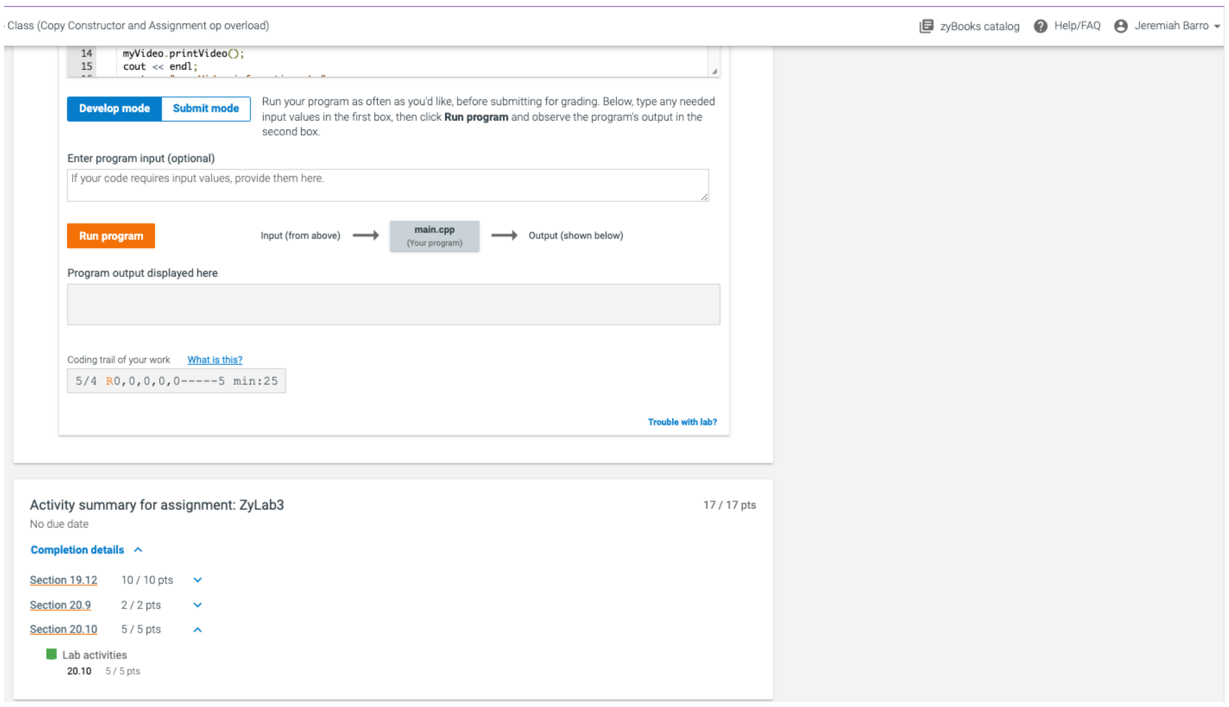


CS 162: Computer Science II
Expires Jul 2nd, 2023

Table of contents / About this material	zyLabs	Challenge	Participation
19. CS 162 Dynamic Variables and Memory Manag...	100%	100%	100%
20. CS 162 Dynamic Variables, Rule of Three - Part II	100%	100%	100%

View my activity
Select date and time below to show all activity up until a specific time. Default is current time.
May 4th, 2023 | 11:59 pm | PDT

Assigned zyLabs completion screenshot:


Class (Copy Constructor and Assignment op overload)

zyBooks catalog · Help/FAQ · Jeremiah Barro

```
    myVideo.printVideo();
    cout << endl;
```

Develop mode | Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program

Input (from above) → main.cpp (Your program) → Output (shown below)

Program output displayed here

Coding trail of your work What is this?

```
5/4 R0,0,0,0,0-----5 min:25
```

Trouble with lab?

Activity summary for assignment: ZyLab3 — 17 / 17 pts

No due date

Completion details

Section 19.12	10 / 10 pts
Section 20.9	2 / 2 pts
Section 20.10	5 / 5 pts

🟩 Lab activities

20.10 5 / 5 pts

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 is console app for creating, deleting, searching, and viewing activities. It stores the name, location, type, rating, and level of each activity. The user can search for stored activities by type, location or name. Data is read from a local file and stores changes when the application quits.

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!
Please select a menu option:
1. Add activity
2. Remove activity
3. Search for activity by name
4. Search for activity by location
5. Search for activity by type
6. Show all activities
7. Quit

1
What’s the name of the new activity: Some new activity
What’s the location? Some place
What’s the type of activity? Food
What’s the rating? Four stars
What’s the level? 5

Here’s your activities:

Some new activity;Some place;Food;Four stars;5

Please select a menu option:
1. Add activity
2. Remove activity
3. Search for activity by name
4. Search for activity by location
5. Search for activity by type
6. Show all activities
7. Quit

2
Which activity would you like to remove?
1
Here’s your activities:

Please select a menu option:
1. Add activity
2. Remove activity
3. Search for activity by name
4. Search for activity by location
5. Search for activity by type
6. Show all activities
7. Quit
```

```
1
What’s the name of the new activity: Some new activity
What’s the location? Some place
What’s the type of activity? Food
What’s the rating? Four stars
What’s the level? 5

Here’s your activities:

Some new activity;Some place;Food;Four stars;5

Please select a menu option:
1. Add activity
2. Remove activity
3. Search for activity by name
4. Search for activity by location
5. Search for activity by type
6. Show all activities
7. Quit

3
What’s the name of the activity you would like to search for? Some new activity

Here’s your activities:

Some new activity;Some place;Food;Four stars;5

Please select a menu option:
1. Add activity
2. Remove activity
3. Search for activity by name
4. Search for activity by location
5. Search for activity by type
6. Show all activities
7. Quit

4
What’s the location of the activity you would like to search for? Some place

Here’s your activities:

Some new activity;Some place;Food;Four stars;5
```

```
Please select a menu option:
1. Add activity
2. Remove activity
3. Search for activity by name
4. Search for activity by location
5. Search for activity by type
6. Show all activities
7. Quit

5
What’s the name of the activity you would like to search for? Some new activity

Here’s your activities:

Some new activity;Some place;Food;Four stars;5

Please select a menu option:
1. Add activity
2. Remove activity
3. Search for activity by name
4. Search for activity by location
5. Search for activity by type
6. Show all activities
7. Quit

6
What’s the type of the activity you would like to search for? Food

Here’s your activities:

Some new activity;Some place;Food;Four stars;5

Please select a menu option:
1. Add activity
2. Remove activity
3. Search for activity by name
4. Search for activity by location
5. Search for activity by type
6. Show all activities
7. Quit
```

```
7
Goodbye
```

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.
Class Activity will have: char array for name   char array for location   char array for level   integer for rating   Enum for types. Class ActivityList will have: array for list of Activity objects held in ActivityList   int for size of ActivityList Main will have: char array for filename   ActivityList class for activityList   char for userInput
b. Identify and list all of the user output and their data types.
Class Activity will have: char array for name   char array for location   char array for level   integer for rating   Enum for types. Class ActivityList will have: array for list of Activity objects held in ActivityList   int for size of ActivityList

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.

Loops for search functionality. Each search will simply print the activity details if an input string matches data of an activity.

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 char userInput
DECLARE ifstream inFile
DECLARE ofstream outFile
DECLARE Activity activities[30]

Load data from local file
WHILE userInput != q
Display menu()
INPUT userInput()
ExecuteInput(userInput)
END
Write Data in memory to file

FUNCTION void ExecuteInput(userInput)
SELECT userInput
   CASE a: DISPLAY  Add activity menu
   CASE b: DISPLAY Remove activity menu
   CASE c: DISPLAY Search for activity by name menu
   CASE d: DISPLAY Search for activity by location menu
   CASE e: DISPLAY Search for activity by type menu
   CASE f: DISPLAY Show all activities
   CASE q: DISPLAY Quit program
   DEFAULT: DISPLAY "Please enter valid option"
END SELECT
END FUNCTION

FUNCTION void name AddActivity(activity)
   Get all inputs required for activity struct
   Print all activities to screen
END FUNCTION

FUNCTION void name RemoveActivity(activities, numActivities)
   Get index of item to remove
   Move all items in array back one space and reduce size of array by one
```

```
  DISPLAY All activities
END FUNCTION

FUNCTION void name SearchActivityByName(activities, numActivities)
  Get name of activities from user to search for
  Loop through activities
    If activity matches name from user input print it
END FUNCTION

FUNCTION void name SearchActivityByLocation(activities, numActivities)
  Get location of activities from user to search for
  Loop through activities
    If activity matches location from user input print it
END FUNCTION

FUNCTION void name SearchActivityByType(activities, numActivities)
  Get type of activities from user to search for
  Loop through activities
    If activity matches type from user input print it
END FUNCTION
```

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
		`IF num_dogs > 10 THEN`
		`     DISPLAY "You have more than 10 dogs!"`
		`ELSE`
		`     DISPLAY "You have ten or fewer dogs!"`

	END IF	`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`