CS 161A: Programming and Problem Solving I

Assignment xx Algorithmic 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.

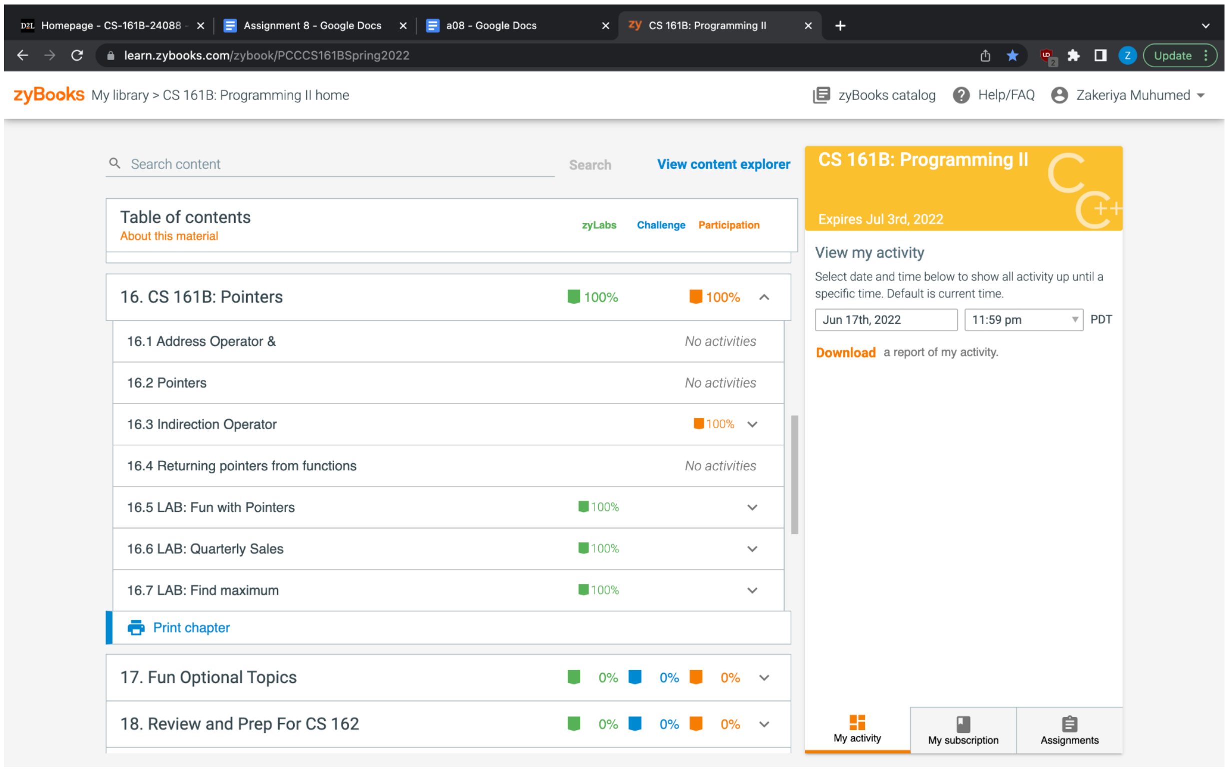
Planning your program before you start coding is part of the development process. In this document you will:

- ❏ 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:


Homepage - CS-161B-24088 | Assignment 8 - Google Docs | a08 - Google Docs | CS 161B: Programming II

learn.zybooks.com/zybook/PCCCS161BSpring2022

zyBooks My library > CS 161B: Programming II home

zyBooks catalog | Help/FAQ | Zakeriya Muhumed

Search content | Search | View content explorer

Table of contents
About this material

	zyLabs	Challenge	Participation
16. CS 161B: Pointers	🟩 100%		🟧 100%
16.1 Address Operator &	No activities		
16.2 Pointers	No activities		
16.3 Indirection Operator			🟧 100%
16.4 Returning pointers from functions	No activities		
16.5 LAB: Fun with Pointers	🟩 100%		
16.6 LAB: Quarterly Sales	🟩 100%		
16.7 LAB: Find maximum	🟩 100%		
17. Fun Optional Topics	🟩 0%	🟦 0%	🟧 0%
18. Review and Prep For CS 162	🟩 0%	🟦 0%	🟧 0%

Print chapter

CS 161B: Programming II
Expires Jul 3rd, 2022

View my activity
Select date and time below to show all activity up until a specific time. Default is current time.
Jun 17th, 2022 | 11:59 pm | PDT
Download a report of my activity.

My activity | My subscription | Assignments

Assigned zyLabs completion screenshot:

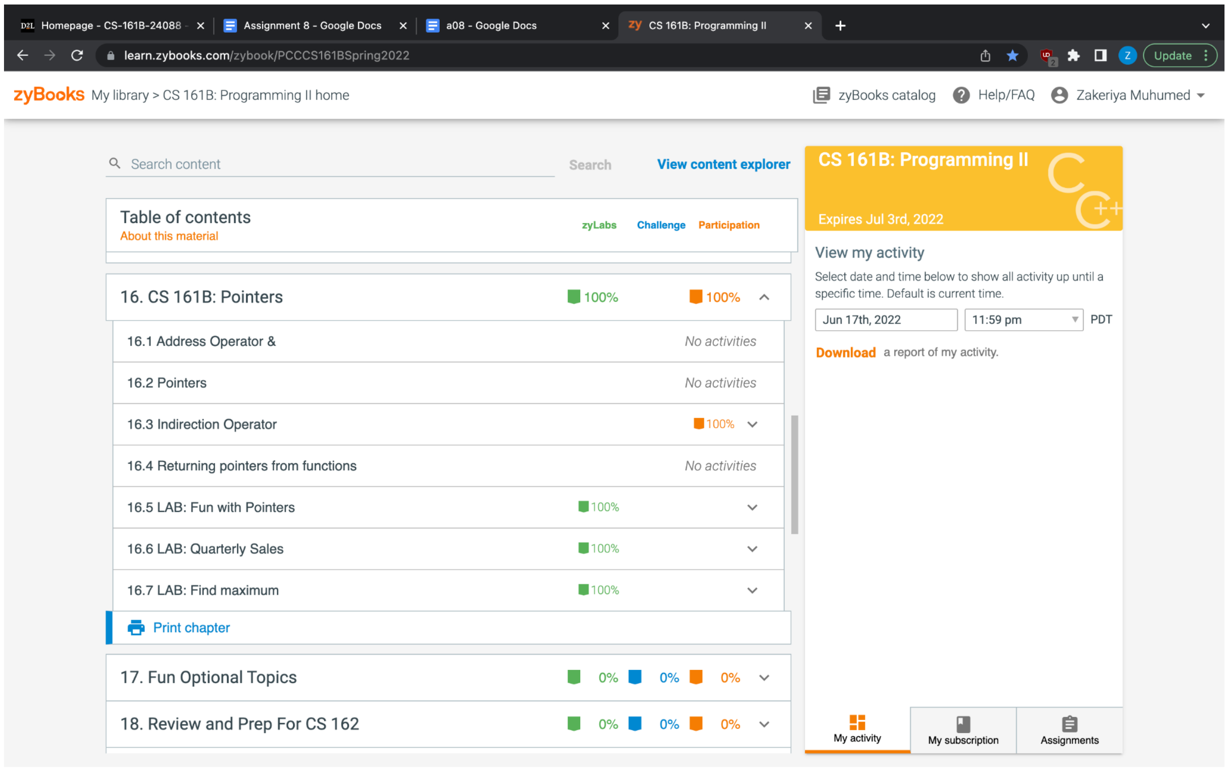

zyBooks My library > CS 161B: Programming II home

Table of contents
About this material

	zyLabs	Challenge	Participation
16. CS 161B: Pointers	🟩 100%		🟧 100%
16.1 Address Operator &	No activities		
16.2 Pointers	No activities		
16.3 Indirection Operator			🟧 100%
16.4 Returning pointers from functions	No activities		
16.5 LAB: Fun with Pointers	🟩 100%		
16.6 LAB: Quarterly Sales	🟩 100%		
16.7 LAB: Find maximum	🟩 100%		
17. Fun Optional Topics	🟩 0%	🟦 0%	🟧 0%
18. Review and Prep For CS 162	🟩 0%	🟦 0%	🟧 0%

CS 161B: Programming II
Expires Jul 3rd, 2022

View my activity
Select date and time below to show all activity up until a specific time. Default is current time.
Jun 17th, 2022 11:59 pm PDT
Download a report of my activity.

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:
In this assignment, you will create a single source code file that uses pointer variables and functions.

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:

```
Enter integer 1: 3
Enter integer 2: 17

Before call to swapArgs a: 3 b: 17
After call to swapArgs a: 17 b: 3
After call to divideArgs a: 5 b: 2
After call to powerArgs a: 25 b: 2

Goodbye!
```

```
Enter integer 1: 2
Enter integer 2: 10

Before call to swapArgs a: 2 b: 10
After call to swapArgs a: 10 b: 2
After call to divideArgs a: 5 b: 0
After call to powerArgs a: 1 b: 0

Goodbye!
```

```
Enter integer 1: -10
Enter integer 2: 3

Before call to swapArgs a: -10 b: 3
After call to swapArgs a: 3 b: -10
After call to divideArgs a: 0 b: 3
After call to powerArgs a: 0 b: 3

Goodbye!
```

```
Enter integer 1: 0
Enter integer 2: 0

No operations performed!
```

```
Enter integer 1: -2
Enter integer 2: -9

Before call to swapArgs a: -2 b: -9
After call to swapArgs a: -9 b: -2
After call to divideArgs a: 4 b: -1
After call to powerArgs a: 4 b: -1

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.
● *x as int pointer data type stores the integer 1 ● *y as int pointer data type stores the integer 2
b. Identify and list all of the user output and their data types.
Modify *x and *y. Such as these 2 variable swapped, divide, and powered
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.
Addiction, Multiply and a for loop for the power.
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.

```
1. FUNCTION swapArgs
    a. DECLARE int tempVal to *x
    b. SET *x to *y
    c. SET *y to *x

   END OF FUNCTION

2. FUNCTION divideArgs
    a. DECLARE int temp1 to *x
    b. DECLARE int temp2 to *y
    c. SET *x to temp1/temp2
    d. SET *y to temp2 % temp1
    END OF FUNCTION

3.  FUNCTION powerArgs
        a.  DECLARE  int tempVal =  *x
        b.  IF *y == 0
                i.    SET *x to 1
        c.  ELSE IF *y < 0
                i.    FOR int i = 0 til *y-1, increase i ++
                          1.  SET *x =  1 / (*x*tempVal);
                ii.   END OF FOR LOOP/ ELSE IF
        d.  ELSE
        e.  FOR int i = 0 til *y-1, increase i ++
                i.    SET *x *= tempVal;
        f.  END OF ELSE

    END OF FUNCTION

4.  FUNCTION main()
        a.  DECLARE *x to new int
        b.  DECLARE *y to new int
        c.  DISPLAY  “Enter integer 1”
        d.  INPUT  *x
        e.  DISPLAY  “Enter integer 2”
        f.  INPUT  *y
        g.  IF *x == *y
                i.    DISPLAY  No operations performed!
                ii.   RETURN 0
        h.  DISPLAY Before call to swapArgs a:  variable  *x   b: variable  *y
        i.  CALL swapArgs
        j.  DISPLAY  After call to swapArgs a:  variable  *x   b: variable  *y
        k.  CALL divideArgs
        l.  DISPLAY After call to divideArgs a:  variable  *x   b: variable  *y
        m.  CALL powerArgs
        n.  DISPLAY After call to owerArgs  a:  variable  *x   b: variable  *y
        o.  DISPLAY “goodbye”
        p.  RETURN 0

    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 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`