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.

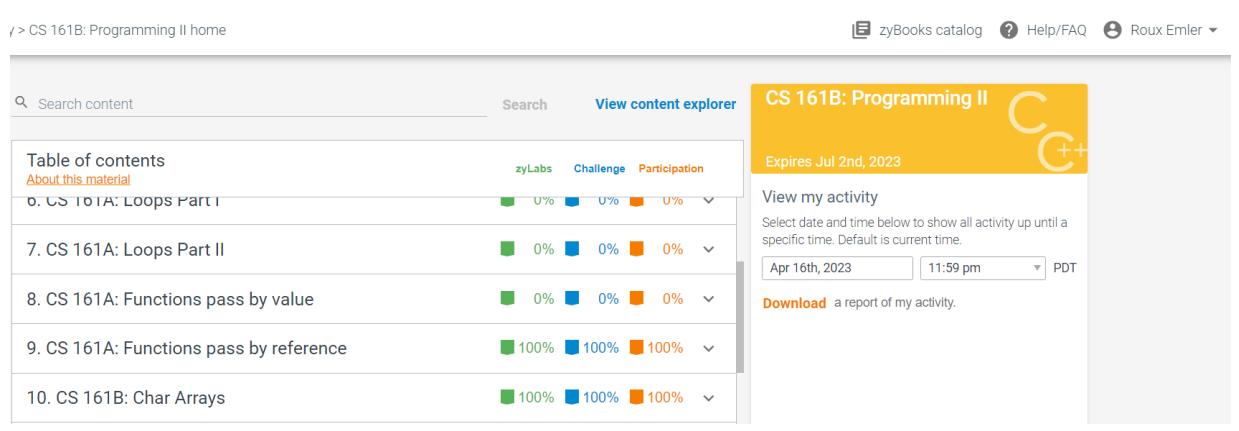
Planning your program before you start coding is part of the development process. In this document you will:

- ❏ 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

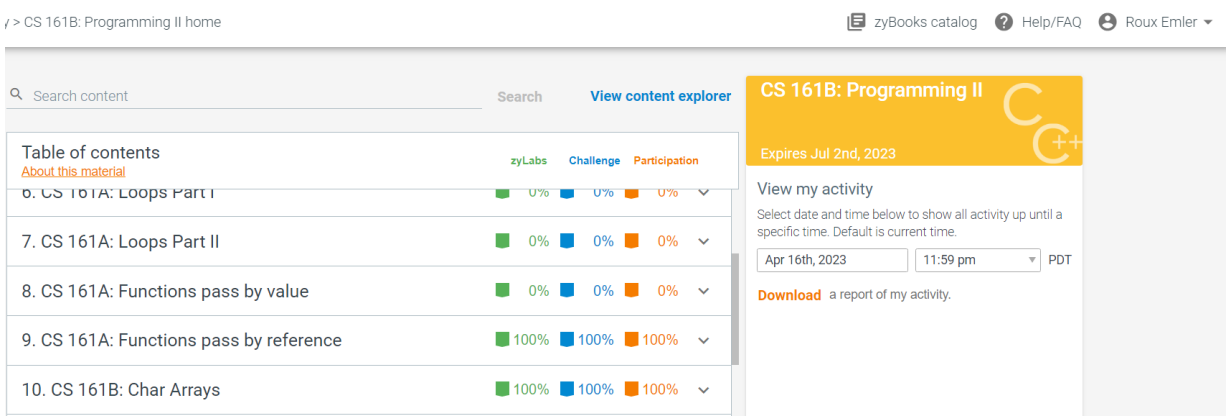
1. zyBooks

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:



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 takes information about the user and the name of a file. This information is than used to encode a more descriptive file name which is then output to the user.

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 to my fileName encoding program!!

Please pick an option below:
(e)Encode a file name
(q)quit
>> e
```

```
This program will ask you a few questions and generate an
encoded fileName based on your answers.

Enter your last name: Iyer

Enter your first name: GD

Was your assignment Late (y/n)? Y

Enter your Student-ID (format: 222-22-2222): 234-05-4556

Enter the file name: a05.cpp

() 13:45

Your encoded file name is: iyer_gd_LATE_4556_1345_a05.cpp

Please pick an option below:
(e)Encode a file name
(q)quit
>> b
Invalid option! Please try again!!
Please pick an option below:
(e)Encode a file name
(q)quit
>> q

Thank you for using my fileName generator!
```

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).
userSelection as a character (menu selection - ‘e’/’q’) fName as a character array (first name) lName as a character array (last name) lateYN as a character (is late or not - ‘Y’/’N’) stdID as a character array (student ID) fileName as a character array hours as an integer mins as an integer
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).
INVALID_INPUT as a character array welcomeMessage as a character array instruct as a character array (short instructions) menu as a character array fNamePrompt as a character array lNamePrompt as a character array latePrompt as a character array idPrompt as a character array fileNamePrompt as a character array timePrompt as a character array

encodeFileName as a character array (the encoded file name)

goodbyeMessage as a character array

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.

Use to.lower() to remove capital characters from the names

Use strncpy() to get the last 4 numbers of the ID and store in parsedID

Use strcat() to concatenate the name, id, time, and file name onto the final file

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.

DECLARE MAX_NAME_LENGTH as a const int SET to 30

DECLARE MAX_PARSEDID_LENGTH as a const int SET to 4

DECLARE MAX_ID_LENGTH as a const int SET to 11

DECLARE MAX_FILENAME_LENGTH as a const int SET to 30

DECLARE MAX_TIME_LENGTH as a const int SET to 5

DECLARE MAX_ENCODED_LENGTH as a const int SET to 111

FUNCTION int main()

1. DECLARE userChoice as a char
2. DECLARE encodeFileName as a char array
3. DECLARE quit as a bool SET to false
4. CALL welcomeMessage()
5. WHILE quit is not true
    a. SET userChoice to CALL displayMenu()
    b. IF userChoice is ‘q’

        i. SET quit to true
        ii. CONTINUE
    c. END IF
    d. CALL encode(encodeFileName)
    e. DISPLAY encodeFileName
6. END WHILE
7. DISPLAY goodbyeMessage

END FUNCTION

FUNCTION void welcome()

1. DECLARE welcomeMessage
2. DISPLAY welcomeMessage

END FUNCTION

FUNCTION char displayMenu()

1. DECLARE char option
2. DECLARE char array menu
3. DISPLAY menu
4. INPUT option

END FUNCTION

FUNCTION void encode(char encodeFileName)

1. DECLARE char array fName
2. DECLARE char array lName
3. DECLARE bool lateFlag
4. DECLARE char array parsedID
5. DECLARE char array fileName
6. DECLARE char array strTime
7. CALL readInput(char fName, char lName, bool lateFlag)
8. CALL redInput(char parsedID, char fileName)
9. CALL readTime(char strTime)
10. COPY lName to encodeFileName
11. APPEND fName to encodeFileName
12. IF lateFlag is true
    a. APPEND LATE to encodeFileName
13. END IF
14. APPEND parsedID to encodeFileName
15. APPEND strTime to encodeFileName
16. APPEND fileName to encodeFileName

17.

FUNCTION void readInput(char fName[], char lName[], bool& lateFlag)

1. DECLARE char lateChar
2. DISPLAY lNamePrompt
3. INPUT lName
4. CALL makeLower(lName)
5. DISPLAY fNamePrompt
6. INPUT fName
7. CALL makeLower(fName)
8. DO
    a. DISPLAY latePrompt
    b. INPUT lateChar
9. WHILE lateChar doesn't equal 'Y' or 'N'
10. END DO WHILE
11. IF latePrompt is equal to 'Y'
    a. SET lateFlag to true
12. ELSE
    a. SET lateFlag to false
13. END IF

END FUNCTION

FUNCTION void readInput(char parsedID[], char fileName[])

1. DECLARE stdID as a char array
2. DISPLAY idPrompt
3. INPUT stdID
4. SET parsedID to last 4 digits of userID
5. DISPLAY fileNamePrompt
6. INPUT fileName

END FUNCTION

FUNCTION void readTIme(char strTime[])

1. DECLARE int hours
2. DECLARE int mins
3. DECLARE char discard
4. DO
    a. IF there was an input error or discard is not equal to ':'
        i. DISPLAY INVALID_INPUT
    b. END IF
    c. DISPLAY timePrompt
    d. INPUT hours

e. INPUT discard
f. INPUT mins

5. WHILE discard is not equal to ':' or there was an input error
6. COPY hours to timeStr
7. COPY mins to timeStr

END FUNCTION

FUNCTION makeLower(char name[])

1. FOR the length of name
    a. SET each letter to lowercase
2. END FOR

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`