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:

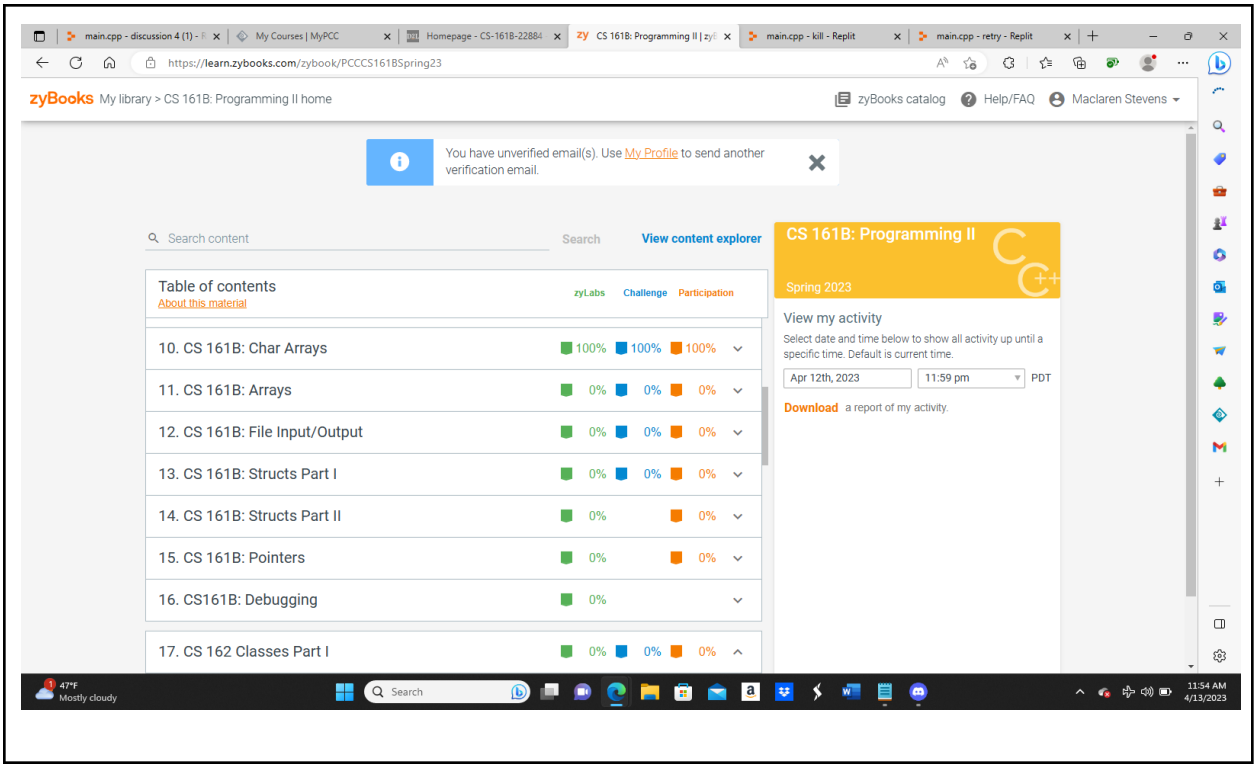
- ❏ Paste a screenshot of your zyBooks Challenge and Participation %
- ❏ Paste a screenshot of your assigned zyLabs completion
- ❏ Write a detailed description of your program, at least two complete sentences
- ❏ If applicable, design a sample run with test input and output
- ❏ Identify the program inputs 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:
Creating an encoded file name out of the users first name, last name, if it is late or not, file name, what time has been submitted, and the users student ID.

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

>> d

Invalid input. Please try again.

>> e

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

Enter your last name: SteVens

Enter your first name: MaC

Was your assignment Late (y/n)? y

Enter your Student-ID (format: 222-22-2222): 234-54-9876

Enter the name of the file: assignment2.cpp

Enter the time submitted (military time - ex: 18:24 for 6:24pm): 12:23

Encoded file name: stevens_mac_LATE_9876_1223_assignment2.cpp

Please pick an option below:

(e) Encode a file name

(q) Quit

>> q

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.

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).
Char menuOption, fName, lName, lateOption, stdID, colon, Int hour, minute
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).
I was confused about what to put for this part.
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.

strTime[0] = (hour / 10) + '0'

strTime[1] = (hour % 10) + '0'

strTime[2] = (minute / 10) + '0'

strTime[3] = (minute % 10) + '0'

strTime[4] = '\0'

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 Void function welcome()

Declare Char function displayMenu()

Declare void function readInput with parameters(char fName[], char lName[], bool& lateFlag)

Declare void function readInput with parameters (char parsedID[], char fileName[])

Declare void function readTime with parameter (char strTime[])

Declare void function encode with parameter (char encodeFileName[])

int main()

Declare char menuOption;

Declare char encodeFileName[100]

welcome()

do

menuOption equals displayMenu()

```
        switch (menuOption)
            case 'e':
                encode(encodeFileName);
                Display prompt - "Encoded file name: " encodeFileName
                break;
            case 'q':
                Display prompt - "Goodbye!"
                break
            default:
                Display prompt -  "Invalid input. Please try again."


    while (menuOption is not equal to  'q')
    return 0
Call void welcome()
    Display prompt - "Welcome to my fileName encoding program!!"
char displayMenu()
    char menuOption;
    Display prompt - "Please pick an option below: "
    Display prompt - "(e) Encode a file name"
    Display prompt - "(q) Quit"
    Display prompt -  ">> "
    Input menuOption;
    while (menuOption is not equal to 'e' and menuOption is not equal to 'q')
        Display prompt -  "Invalid input. Please try again."
        Display prompt -  ">> "
```

```
    Input  menuOption
  return menuOption
Call void readInput(char fName[], char lName[], bool& lateFlag)
  Display prompt - "This program will ask you a few questions and generate an encoded fileName based on your answers."
  Display prompt -"Enter your last name: "
  Input lName
  Display prompt - "Enter your first name: "
  Input  fName

  for (int i equal to 0; fName[i] not equal to '\0'; i++)
    fName[i] equal to tolower(fName[i]);

  for (int i equal to 0; lName[i] not equal to '\0'; i++)
    lName[i] equal to tolower(lName[i]);

  Declare char lateOption
  Display prompt - "Was your assignment Late (y/n)? "
  Input lateOption
  while (lateOption is not equal to 'y' and lateOption is not equal to 'n')
    Display prompt-  "Invalid input. Please enter 'y' or 'n'."
    Display prompt - ">> "
    Input lateOption

  lateFlag equal to (lateOption equal to 'y')
```

```
Call void readInput(char parsedID[], char fileName[])
    Display prompt - "Enter your Student-ID (format: 222-22-2222): "
    Declare char stdID[100]
    Input stdID
    strncpy(parsedID, stdID plus 7, 4)

    Display prompt - "Enter the name of the file: "
    Input fileName

Call void readTime(char strTime[])
    Declare int hour, minute
    Declare char colon
    Display prompt-  "Enter the time submitted (military time - ex: 18:24 for 6:24pm): "
    Input hour, colon, minute

    while (hour is less than 0 or hour is greater than 23 or minute is less than 0 or minute is
greater than 59 or colon is not equal to ':')
        Display prompt - "Invalid input. Please enter a valid time in military format."
        Display prompt ">> "
        Input hour, colon, minute

strTime[0] is equal to (hour divided by 10)  plus '0'
    strTime[1] equal to (hour % 10) plus '0'
    strTime[2] equal to (minute divided by 10) plus '0'
    strTime[3] equal to (minute % 10) plus '0'
    strTime[4] equal to '\0'
```

```
Call void encode(char encodeFileName[]) {
    char lName[50], fName[50], parsedID[50], fileName[50], strTime[50]
    Declare bool lateFlag
    readInput(fName, lName, lateFlag);
    readInput(parsedID, fileName);
    readTime(strTime)

    int index equal to 0
    for (int i equal to 0; lName[i] not equal to '\0'; i++) {
        encodeFileName[index] equal to tolower(lName[i])
        index++

    encodeFileName[index] equal to  '_'
    index++;

    for (int i equal to 0; fName[i] not equal to '\0'; i++) {
        encodeFileName[index] equal to tolower(fName[i]);
        index++;

    encodeFileName[index] equal to  '_'
    index++

    if (lateFlag)
        Then encodeFileName[index] equal to 'L'
```

```
    index++;
    encodeFileName[index] equal to 'A'
    index++
    encodeFileName[index] equal to 'T'
    index++
    encodeFileName[index] equal to 'E'
    index++
    encodeFileName[index] equal to '_'
    index++

for (int i equal to 0; parsedID[i] not equal to '\0'; i++)
    encodeFileName[index] equal to parsedID[i]
    index++

encodeFileName[index] equal to '_'
index++

for (int i equal 0; strTime[i] not equal to '\0'; i++)
    encodeFileName[index] equal to strTime[i];
    index++;

encodeFileName[index] equal to '_'
index++

for (int i equal to 0; fileName[i] not equal to '\0'; i++)
```

```
    encodeFileName[index] equal to tolower(fileName[i])
    index++


encodeFileName[index] equal '\0'
```

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
		`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`

	statement DEFAULT: *statement* *statement* 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`