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:

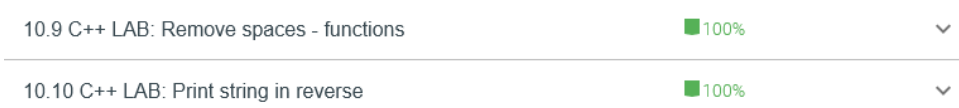
- ❏ Paste a screenshot of your zyBooks Challenge and Participation %
- ❏ Paste a screenshot of your assigned zyLabs completion
- ❏ Write a detailed description of your program, at least two complete sentences
- ❏ If applicable, design a sample run with test input and output
- ❏ Identify the program inputs 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 will output an encoded file name based on user input. Functions will be used to modularize the program.

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: Bittar Enter your first name: Logan Was your assignment Late(y/n)? Y Enter your Student-ID (format: 222-22-2222): 222-22-2222 Enter the file name: a01.cpp Enter the time submitted (military time): 18:24 Your encoded file name is: bittar_logan_LATE_2222_1824_a01.cpp

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).
Char firstName[], first name input Char lastName[], last name input Char stdID[], id input Char strTime[], time input Char fileName[], file name input
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).
Char parsedID[], parsed id of student id Char encodeFileName[], final encoded file name
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.

No calculations needed

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.

FUNCTION void welcome()

FUNCTION char displayMenu()

FUNCTION void encode(char encodeFileName[])

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

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

FUNCTION void toLowerCase(char fName[], char lName[])

FUNCTION void readTime(char strTime[])

1. FUNCTION int main
    a. DECLARE Char display, firstName[], lastName[], parsedID[], fileName[], strTime[], encodeFileName[]
    b. DECLARE Bool late

    c. welcome()
    d. SET display = displayMenu()
    e. IF display == q THEN
        i. DISPLAY Thank you for using my fileName generator
    f. ELSE IF display == e THEN
        i. DISPLAY This program will ask you a few questions and generate an encoded fileName based on your answers.
    g. encode(encodeFileName)
    h. END FUNCTION main

2. FUNCTION void welcome()
    a. DECLARE char display
    b. DISPLAY (e) Encode a file name
    c. DISPLAY (q) quit
    d. INPUT display

    e. WHILE display != q && display != e
        i. DISPLAY Invalid input! Try again!
    f. SET DISPLAY = displayMenu()

    g. RETURN display
    h. END FUNCTION welcome

3. FUNCTION void readInput(char fName[], char lName[], bool &lateFlag)
    a. DECLARE char late
    b. DISPLAY Enter your last name
    c. INPUT lName
    d. DISPLAY Enter your first name
    e. INPUT fName
    f. DISPLAY Was your assignment late(y/n)?
    g. INPUT late
    h. IF late == y OR late == Y
        i. SET lateFlag = true
    i. ELSE IF late == n OR late == N
        i. SET lateFlag = false
    j. WHILE late != y && late != Y && late != n && late != N
        i. DISPLAY INvalid input! Try again!
        ii. DISPLAY Was your assignment late(y/n)?
        iii. INPUT late
    k. END FUNCTION readInput()

4. FUNCTION void toLowerCase(char fName[], char lName[])
    a. FOR fName length
        i. SET fName[i] = tolower(fName[i])
    b. FOR lName length
        i. SET lName[i] = tolower(lName[i])

c. END FUNCTION toLowerCase

5. FUNCTION void readInput(char parsedID[], char fileName[])
    a. DECLARE char stdID[]
    b. DISPLAY Enter your student ID
    c. INPUT stdID
    d. strncpy(parsedID, stdID)
    e. DISPLAY Enter file name:
    f. INPUT fileName
    g. END FUNCTION readInput

6. FUNCTION void readTime(char strTime[])
    a. DECLARE int hour = 0, min = 0
    b. DECLARE char discard
    c. DISPLAY Enter the time submitted (military time):
    d. INPUT hour, discard, min
    e. WHILE !cin OR discard !+ :
        i. DISPLAY Invalid input! Try again!
        ii. INPUT hour, discard, min
    f. strncpy(strTime, to_string(hour.c_str(), 10)
    g. strcat(strTime, to_string(min).c_str())
    h. END FUNCTION readTime

7. FUNCTION void encode(char encodeFileName[])
    a. DECLARE Char display, firstName[], lastName[], parsedID[], fileName[], strTime[], encodeFileName[]
    b. DECLARE Bool late
    c. readInput(firstName, lastName)
    d. toLowerCase(firstName, lastName)
    e. readInput(parsedID, fileName)
    f. readTime(strTime)
    g. IF late == true
        i. strncpy(encodeFileName, lastName, strlen(lastName));
        ii. strcat(encodeFileName, "_")
        iii. strcat(encodeFileName, firstName)
        iv. strcat(encodeFileName, "_")
        v. strcat(encodeFileName, "LATE")
        vi. strcat(encodeFileName, "_")
        vii. strcat(encodeFileName, parsedID)

viii. strcat(encodeFileName, "_")
ix. strcat(encodeFileName, strTime)
x. strcat(encodeFileName, "_")
xi. strcat(encodeFileName, fileName)
xii. DISPLAY Your encoded file name is + encodeFileName
xiii. DISPLAY Thank you for using my fileName generator!

h. ELSE IF late == false
i. strncpy(encodeFileName, lastName, strlen(lastName));
ii. strcat(encodeFileName, "_")
iii. strcat(encodeFileName, firstName)
iv. strcat(encodeFileName, "_")
v. strcat(encodeFileName, parsedID)
vi. strcat(encodeFileName, "_")
vii. strcat(encodeFileName, strTime)
viii. strcat(encodeFileName, "_")
ix. strcat(encodeFileName, fileName)
x. DISPLAY Your encoded file name is + encodeFileName
xi. DISPLAY Thank you for using my fileName generator!

i. END FUNCTION encode()

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*	`IF num_dogs > 10 THEN` `  DISPLAY "You have more than`

	statement
	ELSE
	statement
	statement
	END IF
		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