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:

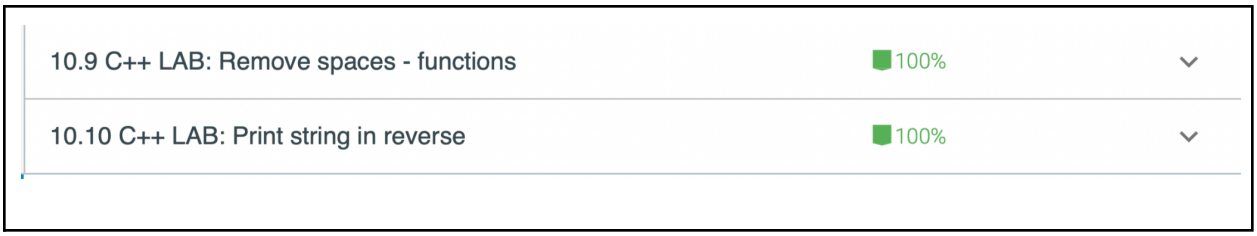
- ❏ Paste a screenshot of your zyBooks Challenge and Participation %
- ❏ Paste a screenshot of your assigned zyLabs completion
- ❏ Write a detailed description of your program, at least two complete sentences
- ❏ If applicable, design a sample run with test input and output
- ❏ Identify the program inputs 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 is a file name encoder. Using several elements from the user, including first name, last name, ID number, and more, the program creates a file based on these things. With the strncpy and strcat functions, the file is encoded.

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

```
Enter your first name: Abby

Enter your last name: Chable

Was your assignment late(y/n): y

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

Enter the file name: a02.cpp

Enter the time submitted (military time - ex: 18:24 for 6:24pm):1:45

Your encoded file name is: chable_abby_LATE_2222_145_a02.cpp

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).
userInput as a character - used for quit or continue fName as an array of characters - users first name lName as an array of characters - users last name yesNo as a character - used for determining if user assignment was late or not stdID as an array of characters - the users ID number fileName as an array of characters - user’s file name hour as an integer - hour entered by the user discard as a character - specifically for “:” in time min as an integer - for minutes
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).
encodeFileName as an array of characters - 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.
strncpy(encodeFileName, lastName, strlen(lastName));

```
  strcat(encodeFileName, firstName);

  strcat(encodeFileName, "_");

strcat(encodeFileName, parID);

  strcat(encodeFileName, time);

  strcat(encodeFileName, nameFile);
```

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 constant integer MAXCHAR and SET to 20

FUNCTION void welcome()

    1. DISPLAY welcome header

FUNCTION character displayMenu()

    1. DECLARE character userInput and SET to ' '
    2. DECLARE boolean validInput and SET to false
    3. WHILE validInput is false
        a. DISPLAY "pick an option from below, (e) encode file or (q) to quit."
        b. INPUT userInput
        c. SET userInput to lowercase userInput
        d. If userInput = 'q'
            i. DISPLAY "Thank you for using my program… bye"
            ii. SET validInput to true
        e. Else if userInput = 'e'
            i. SET validInput to true
        f. ELSE
            i. DISPLAY "Invalid choice"
    4. END WHILE
    5. RETURN userInput

FUNCTION void encode(array of characters encodeFileName)

    1. DECLARE array of characters firstName[MAXCHAR]
    2. DECLARE array of characters lastName[MAXCHAR]
```

3. DECLARE array of characters parID[MAXCHAR]
4. DECLARE array of characters nameFile[MAXCHAR]
5. DECLARE array of characters time[MAXCHAR]
6. DECLARE boolean flag and SET to false
7. CALL readInput(firstName, lastName, flag)
8. CALL readInput(parID, nameFile)
9. CALL readTime(time)
10. CALL strncpy(encodeFileName, lastName, strlen(lastName))
11. CALL strcat(encodeFileName, "_")
12. CALL strcat(encodeFileName, firstName)
13. CALL strcat(encodeFileName, "_")
14. IF flag is false
    a. CALL strcat(encodeFileName, "LATE_")
15. CALL strcat(encodeFileName, parID)
16. CALL strcat(encodeFileName,"_")
17. CALL strcat(encodeFileName, time)
18. CALL strcat(encodeFileName, "_")
19. CALL strcat(encodeFileName, nameFile)
20. DISPLAY “Your encoded filename is encodedFileName”

END FUNCTION

FUNCTION void readInput(array of characters fName, array of characters lName, boolean &lateFlag)

1. DECLARE character yesNo and SET to ‘ ‘
2. DECLARE boolean validationFlag and SET to false
3. DISPLAY “Enter first name: “
4. INPUT fName
5. CALL toLowerCase(fName)
6. DISPLAY “Enter last name: “
7. INPUT lName
8. CALL toLowerCase(lName)
9. WHILE validationFlag is false
    a. DISPLAY “Was your assignment late”
    b. INPUT yesNo
    c. SET yesNo = tolowercase yesNo
    d. IF yesNo = ‘y’
        i. SET lateFlag = false
        ii. SET validationFlag = true
    e. ELSE IF yesNo = ‘n’
        i. SET lateFlag = true
        ii. SET validationFlag = true
    f. ELSE

i. DISPLAY “Invalid Input”
ii. SET validationFlag = false

10. END WHILE

END FUNCTION

FUNCTION void readInput(array of characters parsedID, array of characters fileName)

1. DECLARE array of characters stdID[MAXCHAR]
2. DISPLAY “Enter your ID number: “
3. INPUT stdID
4. CALL strncpy(parsedID, stdID + 7, 4)
5. DISPLAY “Enter file name: “
6. INPUT fileName

END FUNCTION

FUNCTION void readTime(array of characters strTime)

1. DECLARE integer hour and SET to 0
2. DECLARE integer min and SET to 0
3. DECLARE character discard and SET to ‘ ‘
4. DISPLAY “Enter time submitted: “
5. INPUT hour, discard, min
6. WHILE cin is false OR discard does not equal ‘:’ OR 0<hour>24 OR 0<min>60
    a. Display “Invalid Input”
    b. CALL cin.clear
    c. CALL cin.Ignore(100, ‘ ‘)
    d. INPUT hour, discard, min
7. ENDWHILE
8. CALL cin.ignore(100, ‘ ‘)
9. CALL strncpy(strTime, to_string(hour).c_str(),10)
10. CALL strcat(strTime, to_string(min).c_str())

ENDFUNCTION

FUNCTION void toLowerCase(array of characters name)

1. FOR integer i SET to 0, while i is less than length of name, i = i+1
    a. SET name at index i = tolowercase name at index i
2. END FOR

END FUNCTION

FUNCTION main

1. DECLARE array of characters fileName[MAXCHAR x 5]

2. CALL welcome()
3. WHILE displaymenu() does not equal ‘q’
    a. CALL encode(fileName)
4. END WHILE
5. RETURN 0

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
	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`