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.

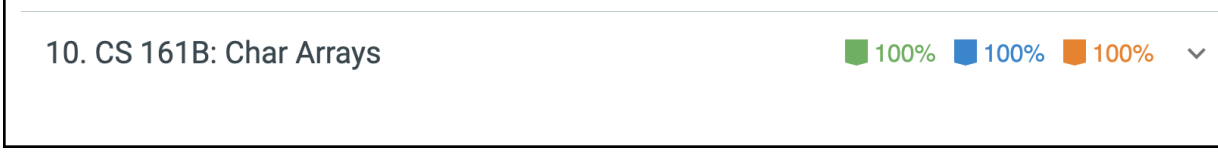
Planning your program before you start coding is part of the development process. In this document you will:

- ❏ 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:

10.9 C++ LAB: Remove spaces - functions	100%
10.10 C++ LAB: Print string in reverse	100%

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 welcomes the user and provides them with options of encoding a file name or quitting the program. The user will then provide their last and first name, if their assignment is late or not, their student ID, file name, and the time they submitted. It will then encode a file name to correlate with the given information and print a goodbye message.

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 >> a Invalid input. Please try again.

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

>> e

Enter your last name: Le

Enter your first name: Brenda

Was your assignment late (y/n)? n

Enter your Student-ID (format: 222-22-2222): 123-45-6789

Enter the file name: a02.cpp

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

Your encoded file name is: le_brenda_6789_1824_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).
- Char array fName → first name - Char array lName → last name - Character late → if the assignment is late or not - Character stdID → student ID - Char array fileName → file name - Integer hour → time submitted (hours) - Integer minute – time submitted (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).
- Char array encodeFileName → 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.

MAXCHAR * 5

stdID + 7, 4

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.

1. DECLARE constant string SPACER and SET to endline
2. DECLARE integer MAXCHAR and SET to 20
3. FUNCTION void welcome()
    a. DISPLAY

        Welcome to my fileName encoding program!

    END FUNCTION welcome()

4. FUNCTION char displayMenu()
    b. DECLARE character array file size SET to MAXCHAR * 5
    c. DECLARE character userChoice and SET to ‘ ‘
    d. DECLLARE boolean validChoice and SET to false
    e. WHILE validChoice is invalid
        i. DISPLAY

            Please pick an option below:

            (e)Encode a file name

            (q)quit

            >>

        ii. INPUT into userChoice
        iii. SET userChoice to lowercase userChoice
        iv. IF userChoice is ‘e’
            1. SET validChoice to true
        v. ELSE IF userChoice is ‘q’
            1. SET validChoice to true
            2. DISPLAY “Thank you for using my fileName generator!”

vi. ELSE
    1. DISPLAY “Invalid input. Please try again.”
vii. END WHILE
viii. RETURN userChoice
ix. END
    1. RESET input stream
    2. CLEAR buffer
    3. INPUT into userNum
x. END WHILE

END FUNCTION displayMenu()

5. FUNCTION void encode(character encodeFileName[])
    f. DECLARE character array firstN file size SET to MAXCHAR
    g. DECLARE character array lastN file size SET to MAXCHAR
    h. DECLARE character array studentID file size SET to MAXCHAR
    i. DECLARE character array time file size SET to MAXCHAR
    j. DECLARE character array fileName file size SET to MAXCHAR
    k. DECLARE boolean flag and SET to false
    l. CALL readInput(firstN, lastN, flag)
    m. CALL readInput(studentID, fileName)
    n. CALL readTime(time)
    o. Copy into string encodeFileName using lastN and string length of lastN
    p. Catenate “_” into encodeFileName
    q. Catenate firstN into encodeFileName
    r. Catenate “_” into encodeFileName
        i. IF flag is false
            1. Catenate “LATE” into encodeFileName
    s. Catenate studentID into encodeFileName
    t. Catenate “_” into encodeFileName
    u. Catenate time into encodeFileName
    v. Catenate “_” into encodeFileName
    w. Catenate fileName into encodeFileName
    x. DISPLAY “Your encoded file name is: “, encodeFileName

END FUNCTION encode()

6. FUNCTION void readInput(char fName[], char lName[], bool &lateFlag)
    y. DECLARE character late and SET to ‘ ‘
    z. DECLARE boolean validFlag and SET to false
    aa. DISPLAY “Enter your last name: “
    bb. INPUT lName
    cc. SET lName to lowercase
    dd. DISPLAY “Enter your first name: “

ee. INPUT fName
ff. SET fName to lowercase
gg. WHILE validFlag is invalid
    i. DISPLAY “Was your assignment late (y/n)?”
    ii. INPUT late
    iii. SET late to lowercase
    iv. IF late is ‘y’
        1. SET lateFlag to false
        2. SET validFlag to true
    v. ELSE IF late is ‘n’
        1. SET lateFlag to true
        2. SET validFlag to true
    vi. ELSE
        1. DISPLAY error message
        2. SET validFlag to false
hh. END WHILE

END FUNCTION readInput()

7. FUNCTION void readInput(char parsedID[], char fileName[])
    ii. DECLARE character array stdID and SET file size to MAXCHAR
    jj. DISPLAY "Enter your Student-ID (format: 222-22-2222): "
    kk. INPUT stdID
    ll. Copy into string parsedID using stdID + 7, 4
    mm. DISPLAY “Enter the file name: “
    nn. INPUT fileName

    END FUNCTION readInput()

8. FUNCTION void readTime(char strTime[])
    a. DECLARE integer hour and SET to 0
    b. DECLARE integer minute and SET to 0
    c. DECLARE character discard
    d. DISPLAY "Enter the time submitted (military time - ex: 18:24 for 6:24pm): "
    e. INPUT hour, discard, minute
    f. WHILE input is invalid OR discard is not “:” OR hour < 0 OR hour > 24 OR min < 0 OR min > 60
        i. DISPLAY error message
        ii. RESET input stream
        iii. CLEAR buffer
        iv. INPUT hour, discard, minute
    g. END WHILE
    h. CLEAR buffer
    i. Copy into strTime using hour

j. Catenate minute into strTime

END FUNCTION readTime()

9. FUNCTION void lowercase(char name[])
    a. FOR integer i is 0, i < string length of name, increment i by 1
    b. SET name at i to lowercase

    END FUNCTION lowercase()

10. FUNCTION main()
    a. DECLARE character file and SET file size to MAXCHAR * 5
    b. CALL welcome()
    c. WHILE displayMenu is 'e'
        i. CALL encode(file)
    d. END WHILE

    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
		`IF num_dogs > 10 THEN`
		`  DISPLAY "You have more than 10 dogs!"`
		`ELSE`
		`  DISPLAY "You have ten or`

	statement END IF	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