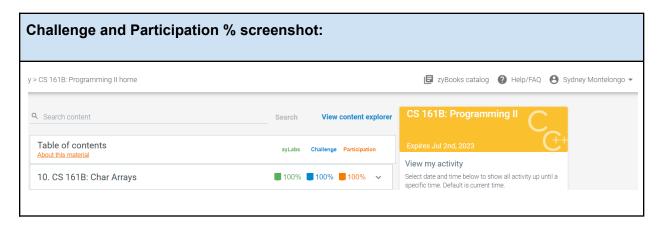
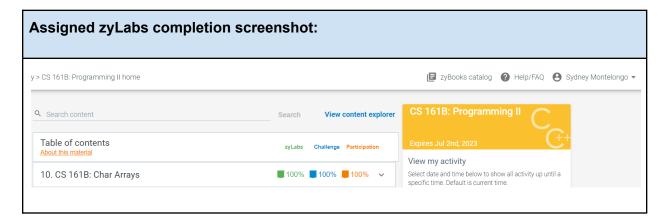
CS 161A/B: Programming and Problem Solving I

Algorithm Design Document

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%.





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 read from the user the student's first and last name, whether or not the assignment was late, their student ID, the file name, and the time submitted. It will use this

information to generate an encoded file name. It will also inloude data validation and a choice menu that allows the user to use the program multiple times.

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
Enter the time submitted (military time - ex: 18:24 for 6:24pm):
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).
 - char userChoice
 - char array fName
 - char array IName
 - char lateResponse
 - char array studentID
 - char array fileName
 - char array time
- 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
- 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.

DECLARE const int MAXCHAR = 50

FUNCTION int main

DECLARE char userChoice

DECLARE char array encodeFileName = ""

CALL welcome

SET userChoice = CALL displayMenu()

WHILE userChoice does not equal 'q'

CALL encode(encodeFileName)

SET userChoice = CALL displayMenu

END WHILE

DISPLAY "Thank you for using my fileName generator!"

END FUNCTION

FUNCTION void welcome

DISPLAY "Welcome to my fileName encoding program!"

END FUNCTION

FUNCTION char displayMenu() DECLARE char userChoice DISPLAY "Please pick an option below: " DISPLAY " (e) Encode a file name DISPLAY " (q) quit DISPLAY ">>" INPUT userChoice SET userChoice to lowercase WHILE userChoice does not equal 'e' and does not equal 'g' DISPLAY "Invalid input! Please try again!!" and clear input stream INPUT userChoice **END WHILE** RETURN userChoice **END FUNCTION** FUNCTION void encode(char encodeFileName[]) DECLARE char array fName

DECLARE char array IName

DECLARE bool lateFlag

DECLARE char array parsedID

DECLARE char array fileName

DECLARE char array strTime

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

CALL readInput(fName, IName, lateFlag)

CALL readInput(parsedID, fileName)

CALL readTime(strTime) SET encodeFileName to format "IName_fName_" IF lateFlag = true THEN SET encodeFileName to format "IName_fName_LATE" **END IF** SET encodeFileName to format "IName_fName_LATE_parsedID_strTime_fileName" (or without late if IF was not taken) DISPLAY "Your encoded file name is: " encodeFileName **END FUNCTION** FUNCTION void readInput(char fName[], char IName[], bool &lateFlag DECLARE char lateResponse DISPLAY "Enter your last name: " **INPUT** name CALL makeLowercase(fName) DISPLAY "Enter your first name: " INPUt fName CALL makeLowercase(IName) DISPLAY "Was your assignment Late?" INPUT lateResponse SET lateResponse to lowercase WHILE lateResponse does not equal 'y' and does not equal 'n' DISPLAY "Invalid response, please try again!" and clear input stream INPUT lateResponse SET lateResponse to lowercase

END WHILE

```
IF lateResponse is 'y' THEN
   SET lateFlag = true
 ELSE IF lateResponse is 'n' THEN
   SET lateFlag = false
 END IF
END FUNCTION
FUNCTION void readInput (char parsedID[], char fileName[])
 DECLARE char studentID[] = "".
 DISPLAY ""Enter your Student-ID (format: 222-22-2222): "
 INPUT studentID
 SET parsedID to the last 4 digits of studentID
 DISPLAY "Enter the file name: "
 INPUT fileName
END FUNCTION
FUNCTION void readTime(char strTime[])
 DECLARE int hour
 DECLARE int min
 DECLARE char discard
 DISPLAY "Enter the time submitted (military time- ex: 18:24 for 6:24 pm): "
 INPUT hour, discard, and min
 WHILE input is not valid, or discard does not = :, or hour is less than zero or greater than
24, or min is less than 0 or greater than 59
   DISPLAY "Invalid input, please try again!" and clear input stream
   INPUT hour, discard, and min
```

END WHILE

SET strTime to contain min and hour

END FUNCTION

FUNCTION void makeLowercase(char userString[])

DELCARE int length

SET length = length of userString

FOR int i = 0 TO i = length

SET userString[i] = tolower(userString[i]

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	<pre>IF num_dogs > 10 THEN DISPLAY "That is a lot of dogs!" END IF</pre>		

Use a dual alternative conditional	IF condition THEN statement statement ELSE statement statement statement	<pre>IF num_dogs > 10 THEN DISPLAY "You have more than 10 dogs!" ELSE DISPLAY "You have ten or fewer dogs!" END IF</pre>		
Use a switch/case statement	SELECT variable or expression CASE value_1: statement statement CASE value_2: statement statement CASE value_2: statement CASE value_1: statement CASE value_2: statement 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	<pre>SET num_dogs = 1 WHILE num_dogs < 10 DISPLAY num_dogs, " dogs!" SET num_dogs = num_dogs + 1 END WHILE</pre>		
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	RETURN value	RETURN 2 + 3
function		