## 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.

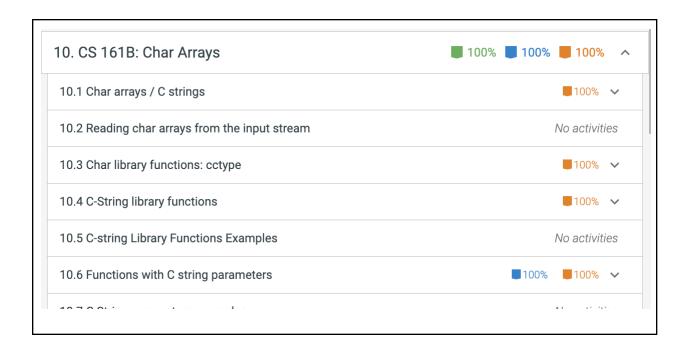
Planning your program before you start coding is part of the development process. In this document you will:

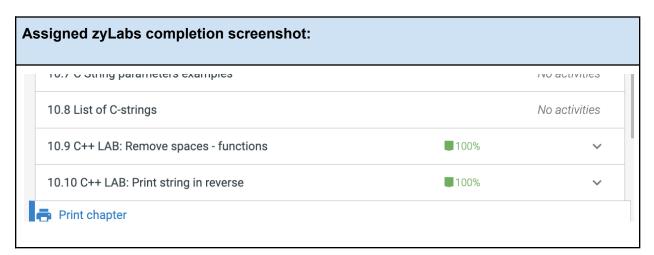
Paste a screenshot of your zyBooks Challenge and Participation %
Paste a screenshot of your assigned zyLabs completion
Write a detailed description of your program, at least two complete sentences
If applicable, design a sample run with test input and output
Identify the program inputs and their data 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:	





### 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 encode a file name for the user, using their first name, last name, student ID #, and if their assignment is late Y/N. The code will use cctype and cstring libraries to manipulate char arrays to create the file name. The user will go through a short menu, and it will loop until the user selects quit.

### 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: Ferguson
Enter your first name: Gina
Was your assignment Late (y/n)? Y
Enter your Student-ID (format: 222-22-2222): 234-05-4556
Enter the file name: a02.cpp
Enter the time submitted (military time - ex: 18:24 for 6:24pm):
13:45
Your encoded file name is: ferguson gina LATE -234 1345 a02.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).

option [char] - used in readOption function to store user's choice from menu options lateInput [char] - used in readInput function to store user's choice if assignment was late hour [integer] - used in readTime function to store what hour of time the user is inputting min [integer] - used in readtime function to store the minutes of the time user is inputting

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).

fileName [char array] - outputs fileName into the encode() function where it gets encoded later

fName [char array] - used to encode file name with first name

IName [char array] - used to encode file name with first name

lateStamp [bool] - used to encode the file with status of assignment late or not

lateInput [char array] - used to encode the file name with status of assignment

lateFlag [bool] - used to put user input to lowercase

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, just loops, if elses, functions, and char array manipulation 😀

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.

#include <iostream>, <cctype>, <cstring>, using namespace std

**FUNCTION** welcome [void] **END FUNCTION** 

FUNCTION displayMenu [void] END FUNCTION

FUNCTION readOption [void] [DECLARE char &option] END FUNCTION

FUNCTION encode [void] [DECLARE char encodeFileName] END FUNCTION

FUNCTION readInput [void] [DECLARE char fName, IName, & bool &lateflag] END FUNCTION

FUNCTION readInput [void] [DECLARE char parsedID, char fileName[]] END FUNCTION

FUNCTION readTime [void] [DECLARE char strTime] END FUNCTION

FUNCTION main [void] [DECLARE integer] END FUNCTION

**DECLARE** option [char], fileName array [50]

CALL welcome

DO

CALL displayMenu

```
CALL readOption (option)
 SELECT option
  CASE = e
  CASE = E
     DISPLAY info statement
     CALL encode(fileName)
     DISPLAY encoded file name message + fileName
  CASE = q
   CASE = Q
      DISPLAY thank you message
   CASE DEFAULT
      DISPLAY Invalid message
  END SELECT
WHILE
  option is NOT q or Q
RETURN 0
END MAIN() FUNCTION
CALL welcome
   DISPLAY welcome message
END FUNCTION CALL
CALL displayMenu
   DISPLAY pick from menu message
END FUNCTION CALL
```

```
CALL readOption [ DECLARE option as ref variable]
  WHILE [true]
      INPUT option
      INPUT ignore to clear buffer
      SET option = lowercase option
      IF option is e or q
         BREAK
      ELSE
        DISPLAY invalid message
      CALL displayMenu
      END IF
END FUNCTION CALL
CALL encode [DECLARE char encodeFileName]
   DECLARE char arrays: fName[50], IName[50], fileName[50], parsedID[5], strTime[5]
   DECLARE boolean lateStamp
  CALL readInput(fName, IName, lateStamp)
  CALL readInput(parsedID, fileName)
  CALL readTime (strTime)
  FOR COUNTER = 0 to null +1
      DECLARE fName[i] as lowercase version
```

```
END FOR
  FOR COUNTER = 0 to null +1
     DECLARE IName[i] as lowercase version
  END FOR
   CALL strcpy(encodeFileName, IName)
   CALL strcat(encodeFileName, "_"
   CALL strcat(encodeFileName, fName)
   IF lateStamp
      CALL strcat(encodeFileName, "_LATE_")
   END IF
   CALL strcat(encodeFileName, "-")
   CALL strcat(encodeFileName, parsedID)
   CALL strcat(encodeFileName, "_"
   CALL strcat(encodeFileName, strTime)
   CALL strcat(encodeFileName, "_"
   CALL strcat(encodeFileName, fileName)
END endcode() FUNCTION
FUNCTION readInput [DECLARE char arrays fName, IName, and boolean ref &lateFlag
   DISPLAY enter last name prompt
   INPUT IName (whole line)
   DISPLAY enter first name prompt
```

INPUT fName (whole line) DECLARE lateInput char DISPLAY assignment late prompt INPUT lateInput lateFlag = lowercase lateInput = 'y' END readInput() FUNCTION #1 FUNCTION OVERLOAD readInput #2 [DECLARE char arrays parsedID, fileName] DISPLAY enter student id prompt INPUT ignore to clear buffer INPUT/GETLINE parsedID + 5 digits + - added INPUT/GETLINE parsedID + 5, 3, + "-" INPUT/GETLINE parsedID + 8, 5 DISPLAY enter file name prompt INPUT/GETLINE (fileName, 50) END readInput() FUNCTION #2 FUNCTION readTime [DECLARE char array strTlme] DECLARE integers hour, min WHILE #1 (true) DISPLAY enter time prompt **INPUT** hour IF input fails, AND hour is less than 0 or more than 23 clear the flag

```
clear the input buffer
         DISPLAY invalid hour message and prompt again
       ELSE
       BREAK
       END IF
    WHILE #2 (true)
        clear the input buffer
        INPUT min
        IF input fails or min is less than 0 or greater than 59
            clear the flag
             clear the input buffer
             DISPLAY invalid minute message
         ELSE
         BREAK
         END IF
      END WHILE #2
CALL sprintf(strTime, "%2d%02d", hour, min)
END WHILE
END readTime() 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:			
	DECLARE	•			
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 &gt; 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 END IF	<pre>IF num_dogs &gt; 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_2:     statement DEFAULT:     statement statement statement 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			

	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 &lt; 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	<pre>SET num_dogs = 1 DO     DISPLAY num_dogs, " dogs!"     SET num_dogs = num_dogs + 1 WHILE num_dogs &lt; 10</pre>				
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				