

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](https://draw.io) - [Diagrams.net](https://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:

10. CS 161B: Char Arrays

100% 100% 100% ▾

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 will request input and information from the user to encode a file name for use in D2L. The requested information will include their first and last name, student ID, file name, punctuality of the assignment, and the submitted time.

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).

- userChoice as a character
- userInput as a character

<ul style="list-style-type: none"> • firstName as character • lastName as character • studentId as character • fileName as character • lateFlag as character • time as character
<p>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).</p>
<ul style="list-style-type: none"> • parsedId as character • encodedFile as a character
<p>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.</p>
<p>N/A</p>
<p>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.</p> <p>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.</p>
<pre>CALL welcome() CALL displayMenu(userInput) CALL encode(encodedFileName) CALL readInput(fName, lName, lastFlag) CALL readInput(parsedID, fileName) CALL readTime(time) FUNCTION main()</pre>

```

DECLARE character userInput
DECLARE encodedFileName
CALL welcome()
SET userInput = displayMenu()
WHILE userInput = 'e'
    CALL encode(encodedFileName)
    DISPLAY "Your encoded file name is: " encodedFileName
    SET userInput = displayMenu()
DISPLAY "Thank you for using my fileName generator!"
END FUNCTION

FUNCTION void welcome()
    DISPLAY "Welcome to my fileName program!"
END FUNCTION

FUNCTION character displayMenu()
    DECLARE userChoice
    DISPLAY "Please pick an option from below:
        (e) Encode a file name
        (q) Quit"
    INPUT userChoice
    WHILE userChoice != e or q
        DISPLAY "Invalid option! PLease try again: "
        INPUT userChoice
    END WHILE

```

```

    RETURN userChoice
END FUNCTION

FUNCTION void encode(encodeFileName)
    DECLARE character fName
    DECLARE character IName
    DECLARE character parsedID
    DECLARE character fileName
    DECLARE character time
    DECLARE boolean lateFlag
    CALL readInput(fName, IName, lateFlag)
    CALL readInput(parsedID, fileName)
    CALL readTime(strTime)

    SET encodedFileName = fName + "_" + IName + "_"

    IF lateFlag == true THEN
        SET encodedFileName += "LATE"
    END IF

    SET encodedFileName += "_" + parsedID + "_" + time + "_" + fileName

```

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	INPUT	INPUT num_dogs

into a variable		
Update the contents of a variable	SET	SET num_dogs = num_dogs + 1
Conditionals		
Use a single alternative conditional	IF <i>condition</i> THEN <i>statement</i> <i>statement</i> END IF	IF num_dogs > 10 THEN DISPLAY "That is a lot of dogs!" END IF
Use a dual alternative conditional	IF <i>condition</i> THEN <i>statement</i> <i>statement</i> ELSE <i>statement</i> <i>statement</i> 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 <i>variable or expression</i> CASE <i>value_1</i> : <i>statement</i> <i>statement</i> CASE <i>value_2</i> : <i>statement</i> <i>statement</i> CASE <i>value_2</i> : <i>statement</i> <i>statement</i> DEFAULT: <i>statement</i> <i>statement</i> 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 <i>condition</i> <i>statement</i> <i>statement</i> 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 <i>statement</i> <i>statement</i> WHILE <i>condition</i>	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 <i>counter</i> = <i>start</i> TO <i>end</i> <i>statement</i> <i>statement</i> END FOR	FOR count = 1 TO 10 DISPLAY num_dogs, " dogs!" END FOR

Functions		
Create a function	FUNCTION <i>return_type</i> <i>name (parameters)</i> <i>statement</i> <i>statement</i> END FUNCTION	<pre> FUNCTION Integer add(Integer num1, Integer num2) DECLARE Integer sum SET sum = num1 + num2 RETURN sum END FUNCTION </pre>
Call a function	CALL <i>function_name</i>	CALL add(2, 3)
Return data from a function	RETURN <i>value</i>	RETURN 2 + 3