

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% ^
10.1 Char arrays / C strings	■ 100% v
10.2 Reading char arrays from the input stream	No activities
10.3 Char library functions: ctype	■ 100% v
10.4 C-String library functions	■ 100% v
10.5 C-string Library Functions Examples	No activities
10.6 Functions with C string parameters	■ 100% ■ 100% v
10.7 C String parameters examples	No activities
10.8 List of C-strings	No activities

Assigned zyLabs completion screenshot:

10. CS 161B: Char Arrays	 100%  100%  100% ^
10.1 Char arrays / C strings	 100% v
10.2 Reading char arrays from the input stream	No activities
10.3 Char library functions: ctype	 100% v
10.4 C-String library functions	 100% v
10.5 C-string Library Functions Examples	No activities
10.6 Functions with C string parameters	 100%  100% v
10.7 C String parameters examples	No activities
10.8 List of C-strings	No activities

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 create a file name for submitted documents based on the user's name, time of submission, student I.D. number and whether the file was submitted late.

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 the fileName generator!

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: Rothstein

Enter your first name: Megan

Was your assignment Late (y/n)? y

Enter your Student-ID (format: 222-22-2222): 123-45-6776

Enter the file name: a02.cpp

Enter the time submitted (military time - ex: 18:24 for 6:24pm):
12:45

Your encoded file name is: Rothstein_Megan_LATE_6776_1245_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 this 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 encodeQuit, fName, IName, parsedID, fileName, strTime, bool lateFlag

- 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 encodeQuit, fName, IName, parsedID, fileName, strTime, bool lateFlag

- 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.

char encodeQuit, fName, IName, parsedID, fileName, strTime, bool lateFlag

- 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 void welcome, void goodbye(), displayMenu(), encode(), readInput(char fName[],  
char IName[], bool &lateFlag), readFile(char parsedID[], char fileName[]), readTime(char  
strTime[]);
```

```
FUNCTION int main()  
  DECLARE char userOption, parsedID[12], fileName[100], strTime[7]  
  CALL welcome  
  SET userOption = displayMenu()  
  IF(userOption == 'e')  
    CALL encode()  
    CALL displayMenu()
```

```

        CALL goodbye()
    END FUNCTION
    FUNCTION void welcome()

        DISPLAY welcome phrase
    FUNCTION char displayMenu()

        DECLARE char encodeQuit;

        DISPLAY message to choose to encode or quit

        READ encodeQuit;

        IF (encodeQuit == 'e')

            DISPLAY message that you are encoding a file name

        ELSE IF (encodeQuit == 'q')

            DISPLAY you selected quit

            END PROGRAM

        ELSE

            DISPLAY message to select e or q or try again

            DISPLAY message to choose to encode or quit

            READ encodeQuit

            IF (encodeQuit != 'e' || encodeQuit != 'q')

                DISPLAY try again message

                RETURN encodeQuit

        FUNCTION void encode()

            DECLARE char fName[20], lName[20], lastFourDigits[5], fileName[100], strTime[7],
            parsedID[20] and bool lateFlag;

            CALL readInput

            CALL readfile

            CALL readTime

```

```

    IF (lateFlag = 'n')
        DISPLAY input info as file name
    IF (lateFlag = 'y')
        DISPLAY input info as file name with late
FUNCTION void readInput(char fName[], char lName[], bool &lateFlag)
    DISPLAY enter name message
    READ fName
    FOR (int i = 0; i < strlen(fName); i++)
        IF (isupper(fName[i]))
            SET fName[i] = tolower(fName[i]);
    DISPLAY enter last name message
    READ lName
    FOR (int i = 0; i < strlen(fName); i++)
        IF (isupper(fName[i]))
            SET fName[i] = tolower(fName[i]);
    DISPLAY did you submit the file late message
    READ lateFlag
    DISPLAY your name is message with fName and lName
    IF lateFlag
        DISPLAY you submitted message
    ELSE
        DISPLAY you submitted late message
FUNCTION void readFile(char parsedID[], char fileName[])
    DECLARE char lastFourDigits[5]
    DISPLAY enter you student ID message

```

```

    READ parsedID

    SET strncpy(lastFourDigits, parsedID + 6, 4)

    SET lastFourDigits[4] = '\0';

    DISPLAY last four digits message

    DISPLAY enter a file name

    READ file name

    DISPLAY your file name

FUNCTION void readTime(char strTime[])

    SET int hour = 0, min = 0, char discard

    DISPLAY time message

    READ hour, discard and min

    WHILE (!cin || discard != ':')

        DISPLAY invalid input message

        READ hour, discard, min

    DISPLAY you entered message

FUNCTION void goodbye()

    DISPLAY goodbye message

```

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	SET	SET num_dogs = num_dogs + 1

variable		
Conditionals		
Use a single alternative conditional	<pre>IF <i>condition</i> THEN <i>statement</i> <i>statement</i> END IF</pre>	<pre>IF num_dogs > 10 THEN DISPLAY "That is a lot of dogs!" END IF</pre>
Use a dual alternative conditional	<pre>IF <i>condition</i> THEN <i>statement</i> <i>statement</i> ELSE <i>statement</i> <i>statement</i> END IF</pre>	<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	<pre>SELECT <i>variable</i> or <i>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</pre>	<pre>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</pre>
Loops		
Loop while a condition is true - the loop body will execute 0 or more times.	<pre>WHILE <i>condition</i> <i>statement</i> <i>statement</i> END WHILE</pre>	<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.	<pre>DO <i>statement</i> <i>statement</i> WHILE <i>condition</i></pre>	<pre>SET num_dogs = 1 DO DISPLAY num_dogs, " dogs!" SET num_dogs = num_dogs + 1 WHILE num_dogs < 10</pre>
Loop a specific number of times.	<pre>FOR <i>counter</i> = <i>start</i> TO <i>end</i> <i>statement</i> <i>statement</i> END FOR</pre>	<pre>FOR count = 1 TO 10 DISPLAY num_dogs, " dogs!" END FOR</pre>
Functions		

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