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.
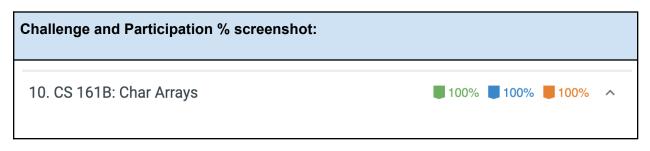
This document contains an interactive checklist. To mark an item as complete, click on the box (the entire list will be highlighted), then right click (the clicked box will only be highlighted), and choose the checkmark.

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:

10. CS 161B: Char Arrays	🟩 100% 🟦 100% 🟧 100% ⌃

Assigned zyLabs completion screenshot:

ZyLab2

20 / 20 pts

No due date

■ 10.9 C++ LAB: Remove spaces - functions 10 / 10 pts
■ 10.10 C++ LAB: Print string in reverse 10 / 10 pts

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 generate an encoded file name after the user answers some question.

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).

Option, fName, lName, fileName, late, ID as char.

hours , mints as integer.

 lateFlag as bool

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).

encodeFileName as char.

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.

Not 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.**

1. FUNCTIONS AND PERIMETERS
    A.   void welcome();
    B.   char displayMenu();
    C.   void encode(char encodeFileName[]);
    D.   void readInput(char fName[], char lName[], bool &lateFlag);
    E.   void readInput(char parsedID[],char fileName[]);
    F.   void readTime(char strTime[]);

2. FUNCTION main()
   a. char encodeFileName[50];
   b. char option;
   c. CALL welcome();
   d. CALL FUNCTION displayMenu();
   e. do while loop
   f. DISPLAY prompt- `Please pick an option below:`
   g. `(e)Encode a file name`
   h. `(q)quit`
   i. INPUT option.
   j. DECLARE option as a char
   k. if (option == 'E' || option == 'e')

      CALL FUNCTION encode(encodeFileName)

   l. DISPLAY- `Your encoded file name is: " encodeFileName`
   m. else if (option == 'q' || option == 'Q')
   n. DISPLAY- `Thank you for using my fileName generator`
   o. return 0;
   p. while (option != 'q' || option != 'Q');
3. CALL FUNCTION void welcome()
   a. DISPLAY– `Welcome to my fileName encoding program!!`

   END FUNCTION welcome()

4. CALL FUNCTION char displayMenu()
   A. DECLARE option as a char
   B. DiSPLAY prompt- Please pick an option below:
   C. (e)Encode a file name
   D. (q)quit
   E. INPUT option

      While

   F. (option != 'E' && option != 'e' && option != 'q' && option !='Q'){
   G. DISPLAY `invalid option! Please try again!!`
   H. INPUT option;
   I. Return option

   END FUNCTION char displayMenu()

5. CALL FUNCTION void encode(char encodeFileName[])
   A. DECLARE fName[] as char
   B. DECLARE lName[] as char
   C. DECLAREparsedID[] as char

D. DECLARE fileName[] as char
E. DECLARE strTime [] as char
F. DECLARE lateFlag as bool
G. readInput(fName,lName,lateFlag);
H. readInput(parsedID,fileName);
I. readTime(strTime);
J. strncpy(encodeFileName, lName , 20);
K. strcat(encodeFileName, "_");
L. strcat(encodeFileName,fName);
M.   strcat(encodeFileName, "_");
N. if (lateFlag)
O.   strcat(encodeFileName, "Late");
P.   strcat(encodeFileName, "_");
Q.   strcat(encodeFileName, parsedID);
R.   strcat(encodeFileName,"_");
S.   strcat(encodeFileName, strTime);
T.   strcat(encodeFileName, "_");
U.   strcat(encodeFileName, fileName);

   END FUNCTION encode(char encodeFileName[])

6. CALL FUNCTION void readTime(char strTime[])
    A. DECLARE hours, mints as integers
    B.   DISPLAY prompt- `Enter the time submitted (military time - ex: 18:24 for 6:24pm):`

    END FUNCTION readTime(char strTime[]

7. CALL FUNCTION void readInput(char fName[], char lName[], bool &lateFlag)
    A. DECLARE late as char
    B. DISPLAY prompt `This program will ask you a few questions and generate an encoded fileName based on your answers.`
    C. DISPLAY prompt- `Enter your last name:`
    D. INPUT lName
    E. DISPLAY prompt- `Enter your first name:`
    F. INPUT fName
    G. DISPLAY prompt- `Was your assignment Late (y/n)?`
    H. INPUT late
    I. while (late != 'Y' && late !='y' && late != 'N' && late !='n')
    J.   DISPLAY prompt `Invalid option! Please try again!!`
    K.   INPUT late
    L.   if (late == 'y' or late == 'Y')
    M.   lateFlag = true;
    N.   else

O.   lateFlag = false;

END FUNCTION readInput(char fName[], char lName[], bool &lateFlag)

8. CALL FUNCTION void readInput(char parsedID[],char fileName[])
    A. DECLARE ID as char
    B. DISPLAY prompt- `Enter your Student-ID (format: 222-22-2222):`
    C. INPUT ID
    D. DISPLAY prompt- `Enter the file name:`
    E. INPUT fileName
    F. strncpy(parsedID, ID +7,4);

END FUNCTION readInput(char parsedID[],char 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 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
		`IF num_dogs > 10 THEN`
		    `DISPLAY "You have more than`

	statement
	ELSE
	   statement
	   statement
	END IF
		```10 dogs!"```
		```ELSE```
		``` DISPLAY "You have ten or```
		```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```