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.

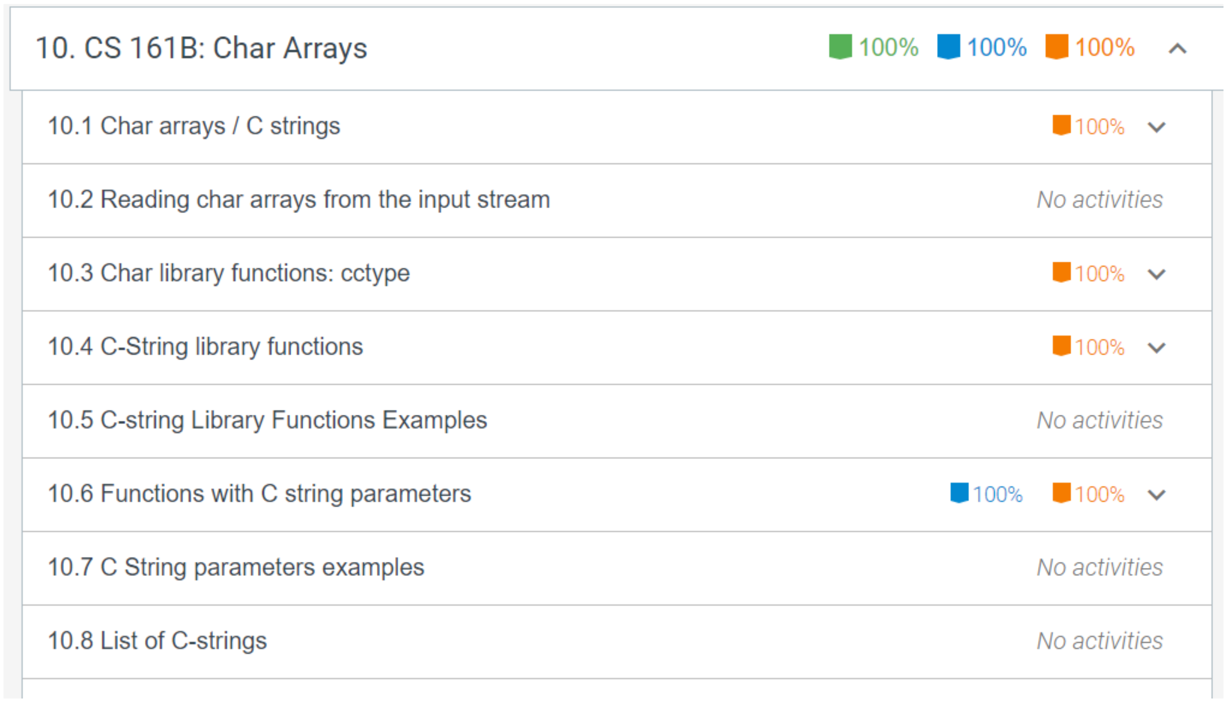
Planning your program before you start coding is part of the development process. In this document you will:

- ❏ 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%.

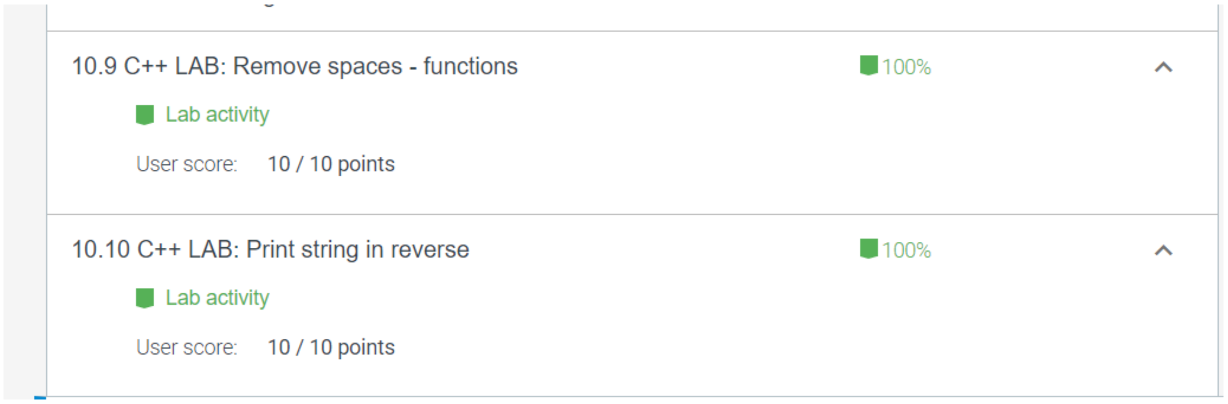
Challenge and Participation % screenshot:



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

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

Assigned zyLabs completion screenshot:



10.9 C++ LAB: Remove spaces - functions 🟩 100%

🟩 Lab activity

User score: 10 / 10 points

10.10 C++ LAB: Print string in reverse 🟩 100%

🟩 Lab activity

User score: 10 / 10 points

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 prompts the user's first and last name, if they're late or not, student ID, file name, and time in order to assemble an encoded file name using the imputed character arrays with the exception of time. Time is imputed as integers representing hours and minutes that will be converted into one character array. The encoded file will be each of the inputs separated by '_' and organized to last name, first name, late or not, last 4 digits of student ID, time, and file name.

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 this encoding program


Please select an option:

(e) to encode

(q) to quit

e

Enter your first name: First

Enter your last name: Name

were you late? (y or n): Y

Please enter your student ID (123-45-6789): 123-45-6789

Please enter the file name (prog.cpp): pop.cpp

Please enter the current time (24:00): 23:00

name_first_LATE_6789_2300_pop.cpp


Please select an option:
```

```
(e) to encode

(q) to quit

q


Thank you for using this encoding program
```

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).

- Fname as a character array
- Lname as a character array
- StudentID as a character array
- FileName as a character array
- Hour as an integer
- Discard as a character
- Min as an integer
- Option as a character
- lateFlag as a character

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 a character array

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.

```
FUNCTION void welcome()
  DISPLAY "Welcome to this encoding program"
END FUNCTION
----------------------------------------------------------
FUNCTION void thankYou()
  DISPLAY "Thank you for using this encoding program"
END FUNCTION
----------------------------------------------------------
FUNCTION character DisplayMenu()
  DECLARE character option

  DO
    DISPLAY "Please select an option: "
    DISPLAY "(e) to encode"
    DISPLAY "(q) to quit"
    INPUT option
    IF (option != 'e' && option != 'q') THEN
      DISPLAY "Invalid input. Please try again."
    END IF
  WHILE (option != 'e' && option != 'q')

  RETURN option;
END FUNCTION
```

```
---------------------------------------------------------
FUNCTION void ReadInput(char fName[], char lName[], bool &lateFlag){
  DECLARE character late
  DECLARE integer i

  DISPLAY "Enter your first name: "
  INPUT fName

  FOR (i = 0 TO strlen(fName)
    SET fName[i] = tolower(fName[i])
  END FOR

  DISPLAY "Enter your last name: "
  INPUT lName

  FOR (i = 0 TO strlen(lName)
    SET lName[i] = tolower(lName[i])
  END FOR

  DO
    DISPLAY "were you late? (y or n): "
    INPUT late;

    SET late = tolower(late)
    IF (late == 'y') THEN
      SET lateFlag = true
    ELSE IF (late == 'n')
      SET lateFlag = false;
    ELSE
      DISPLAY "Invalid input. Please try again." << endl;
    END IF
 WHILE (late != 'y' && late != 'n')
END FUNCTION
---------------------------------------------------------
FUNCTION void ReadInput(char parsedID[], char fileName[])
  DELCARE character studentID[50]
  DELCARE character fileEnd[50]

  DISPLAY "Please enter your student ID (123-45-6789): "
  INPUT studentID
  strncpy(parsedID, studentID + 7, 5)

  DISPLAY "Please enter the file name (prog.cpp): "
  INPUT fileEnd
  strcpy(fileName, fileEnd)
END FUNCTION
-----------------------------------------------------------
FUNCTION void ReadTime(char strTime[])
  DELCARE integer hour = 0
  DELCARE integer min = 0
  DELCARE character discard

  DO
    DISPLAY "Please enter the current time (24:00): "
    INPUT hour
    INPUT discard
    INPUT min

    IF ((hour < 24) && (hour > 0) && (min > 60) && (min < 0)) THEN
      DISPLAY "ERROR. Input is not possible in 24:00 format"
    END IF

  WHILE ((hour > 24) && (hour < 0) && (min > 60) && (min < 0));

  WHILE (!cin || discard != ':')
    DISPLAY "Invalid input! Please try again!!"
    INPUT hour
    INPUT discard
    INPUT min
  END WHILE

  strcpy(strTime, to_string(hour).c_str());
  strcat(strTime, to_string(min).c_str());
  IF (min == 0) THEN
    strcat(strTime, "0")
  END IF
END FUNCTION
-----------------------------------------------------------
FUNCTION void encode()
  DELCARE character encodeFileName[200];
  DELCARE character firstName[20];
  DELCARE character lastName[20];
  DELCARE boolean isLate;
  DELCARE character studentID[50];
  DELCARE character lastPart[50];
  DELCARE character timeString[5];

  CALL ReadInput(firstName, lastName, isLate)
  CALL ReadInput(studentID, lastPart)
  CALL ReadTime(timeString)

  strcpy(encodeFileName, lastName)
  strcat(encodeFileName, "_")
  strcat(encodeFileName, firstName)
  strcat(encodeFileName, "_");
  IF (isLate == true) THEN
    strcat(encodeFileName, "LATE_")
  END IF
  strcat(encodeFileName, studentID)
  strcat(encodeFileName, "_")
  strcat(encodeFileName, timeString)
  strcat(encodeFileName, "_")
  strcat(encodeFileName, lastPart)

  DISPLAY encodeFileName
END FUNCTION
-----------------------------------------------------------
int main()

  CALL welcome();

  WHILE (DisplayMenu() != 'q')
    CALL encode()
  END WHILE

  CALL thankYou();

  RETURN 0;

```

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	`IF num_dogs > 10 THEN`
	statement	`DISPLAY “That is a lot of dogs!”`
	statement	`END IF`
	END IF	
Use a dual alternative conditional	IF *condition* THEN	`IF num_dogs > 10 THEN`
	statement	`DISPLAY “You have more than 10 dogs!”`
	statement	`ELSE`
	ELSE	`DISPLAY “You have ten or fewer dogs!”`
	statement	`END IF`
	statement	
	END IF	
Use a switch/case statement	SELECT *variable or expression*	`SELECT num_dogs`
	CASE *value_1:*	`CASE 0: DISPLAY “No dogs!”`
	statement	`CASE 1: DISPLAY “One dog..”`
	statement	`CASE 2: DISPLAY “Two dogs..”`
	CASE *value_2:*	`CASE 3: DISPLAY “Three dogs..”`
	statement	`DEFAULT: DISPLAY “Lots of dogs!”`
	statement	`END SELECT`
	CASE *value_2:*	
	statement	
	statement	
	DEFAULT:	
	statement	
	statement	
	END SELECT	
Loops		
Loop while a condition is true - the loop body will execute 0 or more times.	WHILE *condition*	`SET num_dogs = 1`
	statement	`WHILE num_dogs < 10`
	statement	`DISPLAY num_dogs, “ dogs!”`
	END WHILE	`SET num_dogs = num_dogs + 1`
		`END WHILE`
Loop while a condition is true - the loop body will execute 1 or more times.	DO	`SET num_dogs = 1`
	statement	`DO`
	statement	`DISPLAY num_dogs, “ dogs!”`
	WHILE *condition*	`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`