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.

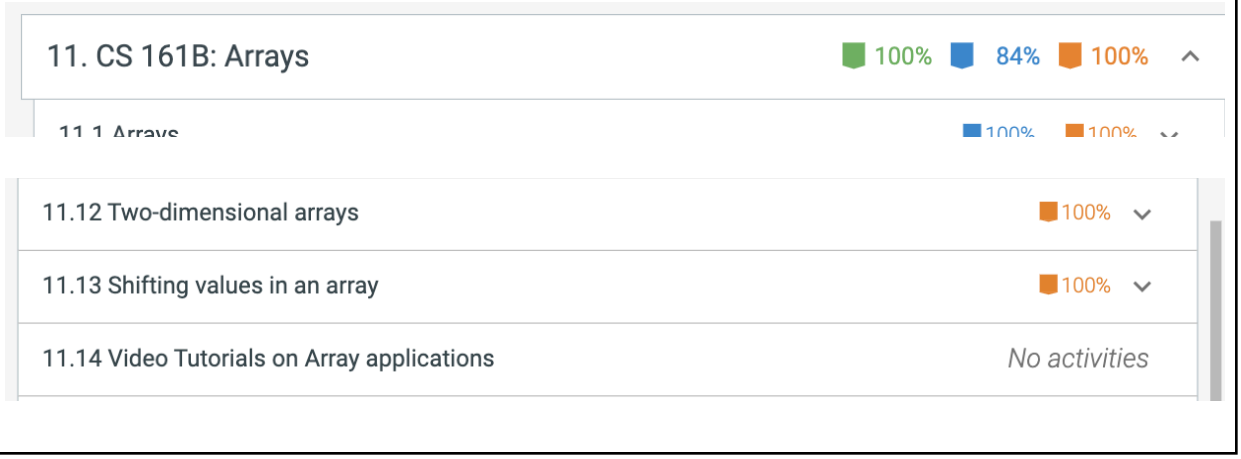
Planning your program before you start coding is part of the development process. In this document you will:

- [x] ~~Paste a screenshot of your zyBooks Challenge and Participation %~~
- [x] ~~Paste a screenshot of your assigned zyLabs completion~~
- [x] ~~Write a detailed description of your program, at least two complete sentences~~
- [x] ~~If applicable, design a sample run with test input and output~~
- [x] ~~Identify the program inputs and their data types~~
- [x] ~~Identify the program outputs and their data types~~
- [x] ~~Identify any calculations or formulas needed~~
- [x] ~~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:



Assigned zyLabs completion 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 is a program for reading course numbers, number of students enrolled, and from user input. Then the program will cancel any courses that have < 10 student and remove them from the courses list, and shift the array to do so. It will also sort by The maximum number of students is <= 25, and the maximum number of courses is 20, with maximum characters of 51.

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 Course Rosters program!!
Enter course number and students enrolled when prompted.
Enter Quit or quit for course number when you are done.
Enter course number: CS133G
```

```
Number of students enrolled: 7
Enter course number: CS160
Number of students enrolled: 15
Enter course number: CS161A
Number of students enrolled: 21
Enter course number: CS161B
Number of students enrolled: 23
Enter course number: CS162
Number of students enrolled: 30
Invalid number!! Please enter a number between 0 and 25
Number of students enrolled: 25
Enter course number: CS260
Number of students enrolled: 5
Enter course number: Quit

List of courses and students:
CS133G          7
CS160           15
CS161A          21
CS161B          23
CS162           25
CS260           5

List after cancellations:
CS160           15
CS161A          21
CS161B          23
CS162           25


Thank you for checking out my Course Rosters program!
```

```
Welcome to my Course Rosters program!!
Enter course number and students enrolled when prompted.
Enter Quit or quit for course number when you are done.
Enter course number: CS133G
Number of students enrolled: 7
Enter course number: CS160_1
Number of students enrolled: 15
Enter course number: CS161A_1
Number of students enrolled: 21
```

```
Enter course number: CS161B
Number of students enrolled: 23
Enter course number: CS162
Number of students enrolled: 30
Invalid number!! Please enter a number between 0 and 25
Number of students enrolled: 25
Enter course number: CS260
Number of students enrolled: 5
Enter course number: CS160_2
Number of students enrolled: 7
Enter course number: CS161A_2
Number of students enrolled: 25
Enter course number: quit

List of courses and students:
CS133G          7
CS160_1         15
CS160_2         7
CS161A_1        21
CS161A_2        25
CS161B          23
CS162           25
CS260           5

List after cancellations:
CS160_1         15
CS161A_1        21
CS161A_2        25
CS161B          23
CS162           25


Thank you for checking out my Course Rosters 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).
● char courseNums[][MAXCHAR]: array storing course numbers input by the user ● int students [] : array storing # of students in each course in put by the user ● int &count: variable storing the count of courses updated as courses are input by the user ● char tempName[MAXCHAR] (both input and output)
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).
● printList(): lists the courses and matching student enrollment # in 2 columns ● welcome() ● char tempName[MAXCHAR] (both input and output)
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.
Code for sorting: ● readInput(): there is a sorting loop that compares and shifts names and student values, it then rearranges them in ascending order based on course names ● cancelCourses(): a loop that iterates through course names and makes sure the number of students is less than 10. If so, the array is shifted and values under 10 are removed.
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
END FUNCTION

FUNCTION void
readInput(char courseNums[][MAXCHAR], int students[], int &count)
END FUNCTION

FUNCTiON void
readInt (string prompt, int &num)
END FUNCTION

FUNCTION void
printList(char courseNums[][MAXCHAR], int students[], int &count)
END FUNCTION

FUNCTION void
cancelCourses(char courseNums[][MAXCHAR], int students[], int &count)
END FUNCTION

FUNCTION int
main
DECLARE courseNums (char) array with MAXCOURSES, & MAXCHAR
DECLARE students (integer) [MAXCOURSES] array
DECLARE count (integer) is 0

CALL welcome function
CALL readInput function: arguments(courseNums, students, count)
DISPLAY list of courses & students text
CALL printList(courseNums, students, count)

CALL cancelCourses function: arguments(courseNums, students, count)

```
DISPLAY list after cancellation text
CALL printList(courseNums, students, count)

DISPLAY thank you message
END PROGRAM
```

// Function Definitions

```
FUNCTION void
   welcome
   DISPLAY “Welcome to my Course Rosters program!!”
   DISPLAY user instruction via string
END FUNCTION

FUNCTION void
   readInt (string prompt, int &num)
   DISPLAY string prompt
   INPUT num

   WHILE input is not in a fail state OR num < 0 OR num > 25,
       DISPLAY invalid error message and re-prompt user
       INPUT clear buffer
       INPUT ignore 100 character spaces, and new line
       DISPLAY prompt
       INPUT num
END FUNCTION

FUNCTiON void
   readInput(char courseNums[][MAXCHAR], int students[], int &count)
      DECLARE char tempName[MAXCHAR] array
      DECLARE i = 0
      DECLARE j = 0

      DISPLAY Enter course number and student # prompt for user
      DISPLAY Enter course number prompt
      INPUT getline → tempName, MAXCHAR
```

```
      WHILE Quit or quit are not being input by user (not equal to each other)

      IF count = CAP
          DISPLAY array is full message
      ELSE
      IF count is not 0
            copy tempName into courseNums[count] (courseNums[count], tempName)
            CALL readInt “Number of students enrolled: “, students[i] (index of students array in
loop)
      ELSE
           WHILE i < count AND COMPARE courseNums[i] to tempName < 0
                iterate LOOP +1
      END IF
      FOR j = count, j > i, j iterates backwards j–
            COPY courseNums[j-1] into courseNums[j]
            SET students at index j equal to students at index j - 1 (to shift left)
       END FOR LOOP
       COPY tempName into courseNums i index (location of loop in the array)
       CALL readInt “Number of students enrolled: “, students[count]
       END 2nd ELSE
       SET count ++ to iterate through loop 1 time each
       IGNORE input 100 spaces and new line
       DISPLAY “Enter course number: “
       INPUT cin.getline tempName, MAXCHAR
       END first ELSE
 END WHILE LOOP
 END FUNCTION

FUNCTION void
   printList(char courseNums[][MAXCHAR], int students[], int &count)
   DECLARE i to 0

    FOR ( i = 0; i < count; i++ )
```

```
        DISPLAY (left-aligned, with a set width of 10) + courseNums[i] + (set width of 10) +
students[i] + new line
      END FOR LOOP
END FUNCTION

FUNCTION void
   cancelCourses(char courseNums[][MAXCHAR], int students[], int &count)
   DECLARE i = 0
   DECLARE j = 0

   FOR i = 0; i < count; i ++
      IF students[i] < 10
         FOR j = i; j < count -1; j++
               COPY courseNums[j+1] into courseNums[j]
               SET students[j] = students[j+1]
         END FOR LOOP #2
      COUNT - - , iterate backwards
      i - -, iterate count index backwards
      END IF
    END FOR LOOP #1
END 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:
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
	  statement
	ELSE
	  statement
	  statement
	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 *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`

My Notes:

Program Description:

- This program uses arrays to read course numbers (a string of text), number of students enrolled (integer) from 2 parallel arrays
- Read the input:
  - readInput(char courseNums[][MAXCHAR], int students[], int &count)
- Cancel courses w/ students < 10:
  - void cancelCourses(char courseNums[][MAXCHAR], int students[], int &count)
- MAX number of students enrolled is <= 25
- MAX number of courses is 20
- MAX number of characters for a course is 51

Program Requirements:

- Shift elements in array(s) to insert or delete elements
- Use a for loop to manipulate the array(s) without going out of bounds
- Pass arrays and their size to a function and manipulate the arrays

readInput (courseNums[ ][MAXCHAR], int students [ ], int &count);

char **courseNums**[ ],[MAXCHAR] <MAXCHAR = 51]

0	1	2
1	CS133G	
2		
3		

int **students**[ ]

0	1	2
1	7	
2		

3		

char **userInput**[MAXCHAR] (51)

0	1	2
1	7	
2		
3		

0	1	2	3
1			
2			
3			
4			
5			
6			