

# 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:**

zyBooks

My library > CS 161B: Programming II home

zyBooks catalog

Help/FAQ

Nonso Agum

Search content

Search

View content explorer

Table of contents

About this material

zyLabs

Challenge

Participation

10. CS 161B: Char Arrays	100%	100%	100%	
11. CS 161B: Arrays	0%	0%	0%	
12. CS 161B: File Input/Output	0%	0%	0%	
13. CS 161B: Structs Part I	0%	0%	0%	
14. CS 161B: Structs Part II	0%		0%	
15. CS 161B: Pointers	0%		0%	
16. CS161B: Debugging	0%			
17. CS 162 Classes Part I	0%	0%	0%	
18. CS 162 Classes Part II	0%	0%	0%	
19. CS 162 Dynamic Variables and Memory Mana...	0%	0%	0%	
20. CS 162 Dynamic Variables, Rule of Three - Par...	0%	0%	0%	
21. CS 162 Dynamic Variables, Debugging - Part III				
22. CS 162 Linked Lists Part I	0%	0%	0%	
23. CS 162 Linked Lists Part II	0%			

CS 161B: Programm

Expires Jul 2nd, 2023

View my activity

Select date and time below to show activity for a specific time. Default is current time.

Apr 13th, 2023

11:

Download a report of my activity

My activity

My subscription

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 program encodes the student's filename using the full name of said student, and their student ID, assignment title/name, (late or early assignment), and timeframe(military 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.



```
Welcome to my fileName encoding program!!

Please pick an option from below:

(e)ncode

(q)uit

>> E

This program will ask you a few questions and generate an encoded
fileName based on your answers.

Enter your last name: Nelson

Enter your first name: Chase

Was your assignment late (y/n)? Y

Enter your Student-ID (format: 222-22-2222): 442-39-9971

Enter the file name: a09.cpp

Enter the time submitted (military time - ex: 18:24 for 6:24pm):
13:12

Your encoded file name is: nelson_chase_LATE_9971_1312_a09.cpp

Pick an option from below
```

```
(e)ncode
```

```
(q)uit
```

```
>> q
```

```
Thank you for using my file name 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 (char) - description: retain user option choice (i.e "q" or "e")

firstname (char) - description: retain user firstname input

lastName (char) - description: retain user firstlast name input

- 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).

parsedID (char) - description: retain user parsed value

fileName (char) - description: retain user assignment fileName input

encodedFileName (char) - description: retain encoded file name

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

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 const int char
- FUNCTION
- void welcome();

void displayMenu();

char readOption();

void readInput(char firstName[], char lastName[], bool &lateFlag);

void readInput(char parsedID[], char fileName[]);

void readTime(char militaryTime[]);

void encode(char encodeFileName[]);

- DECLARE option & encodedfile Name(char)
- FUNCTION welcome() & displayMenu()
- DISPLAY Welcome Message
- FUNCTION void displayMenu(){
- DISPLAY "Please pick an option from below:"

cout << "(e)ncode" << endl;

cout << "(q)uit" << endl;

```

    cout << ">> ";

    • IF while(tolower(option) != 'e' && tolower(option) != 'q') {
    • DISPLAY Error Message "Invalid option, try again!"
    • SET void readInput(char firstName[], char lastName[], bool &lateFlag) {

char lateYN = 0;

    • DISPLAY Last name message
    • SET int i = 0;

for(i = 0; i < strlen(lastName); i++){

    lastName[i] = tolower(lastName[i]);

}

    • DISPLAY Enter First name message
    • SET for(i = 0; i < strlen(firstName); i++){
    •     firstName[i] = tolower(firstName[i]);
    •     }
    • DISPLAY "Was your assignment late (y/n)?"
    • IF USER input is incocrect
    • DISPLAY Invalid option, please try again"
    • IF not SET "if(lateYN == 'y') {
    •     lateFlag = true;
    • }
    • FUNCTION void readInput(char parsedID[], char fileName[]) {
    •     char studentID[MAX] = {0};
    • DISPLAY Studen't ID Message
    • DISPLAY File name Message
    • SET FUNCTION void readTime(char militaryTime[]) {
    •     int hours = 0;
    •     int min = 0;
    •     char discard;
    • DISPLAY "Enter the time submitted (military time - ex: 18:24 for 6:24pm): "
    • WHILE(!cin || hours > 24 || hours < 0) {
    • DISPLAY Invalid Message "Invalid input, please try again!"
    • WHILE while(!cin || min > 60 || min < 0) {
    • DISPLAY Invalid Message
    • IF User provides valid input
    • DISPLAY file name message
    • IF Invalid DISPLAY Error Message
    • IF User Input Valid DISPLAY Encoded File name
    • DISPLAY "pick option from belower to user"
    • IF User answers "q" or "Q"
    • DISPLAY Thank you 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 variable	SET	SET num_dogs = num_dogs + 1
<b>Conditionals</b>		
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> CASE <i>value_2</i> : <i>statement</i>	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!"

	<i>statement</i> CASE <i>value_2</i> : <i>statement</i> <i>statement</i> DEFAULT: <i>statement</i> <i>statement</i> END SELECT	END SELECT
<b>Loops</b>		
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
<b>Functions</b>		
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