# CS 161A: Programming and Problem Solving I

## Assignment A02 Algorithmic 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:

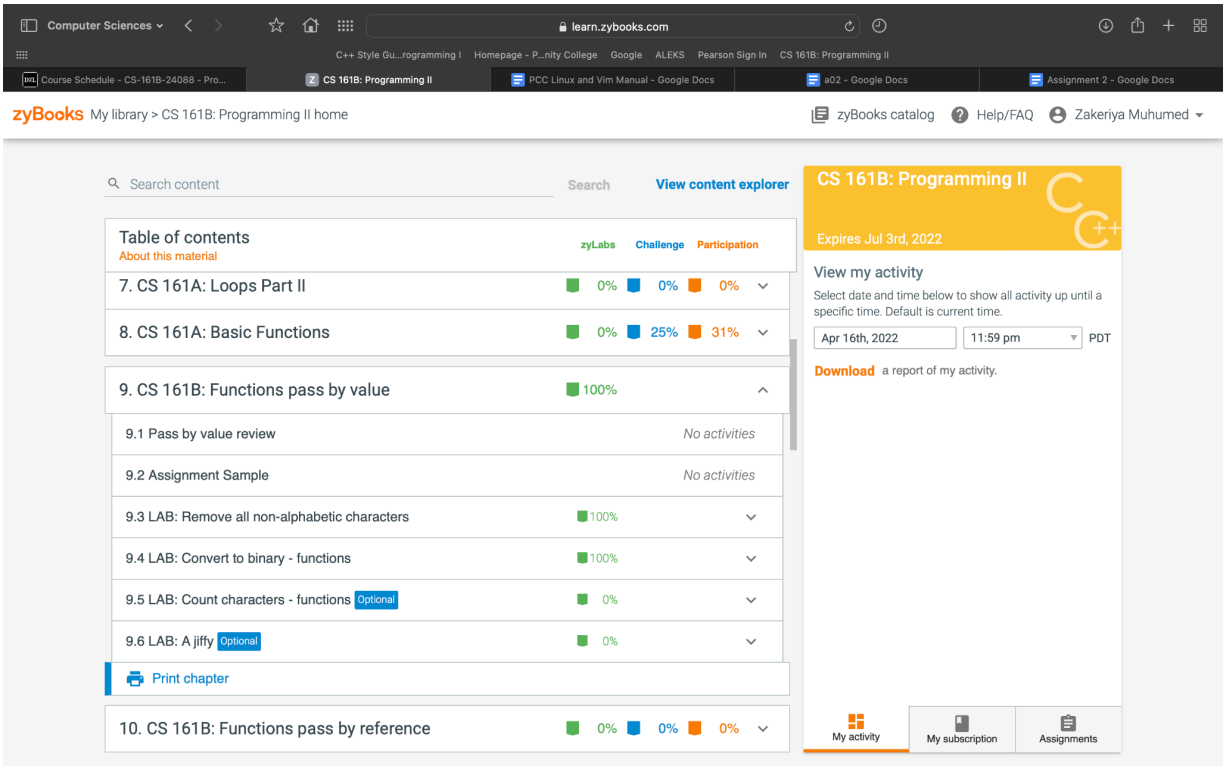
- ❏ Paste a screenshot of your zyBooks Challenge and Participation %
- ❏ Paste a screenshot of your assigned zyLabs completion
- ❏ Write a detailed description of your program, at least two complete sentences
- ❏ If applicable, design a sample run with test input and output
- ❏ Identify the program inputs 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:** |
| --- |
| No challenges and participation today. |

| **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:** |
| --- |
| In this assignment, you will be writing a menu-driven program to do some interest calculations. You will give the user a menu with some choices, and let them pick a choice. Based on the choice they pick, you will ask them some questions and give them results. This process will repeat until they choose to quit the program. The purpose of this assignment is to modularize your program. |
| |

## 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 Interest Calculation program!
What would you like to do today?
Pick an option from below:
        1. Simple Interest
        2. Compound Interest
        3. Quit
>> c
Invalid Option! Please choose 1-3!
>> 9
Invalid Option! Please choose 1-3!
>> 1
Enter the Principal Amount: 300
Enter the interest rate: 3.75
Enter the time in years: 4

Interest accrued: $45.00
Total Accrued Amount (principal + interest): $345.00

What would you like to do today?
Pick an option from below:
        1. Simple Interest
        2. Compound Interest
        3. Quit
>> 2
Enter the Principal Amount: 10000
Enter the interest rate: 3.875
Enter the time in years: 7.5
Enter the number of compounding period: 12

Interest accrued: $3366.37
Total Accrued Amount (principal + interest): $13366.37

What would you like to do today?
Pick an option from below:
        1. Simple Interest
        2. Compound Interest
        3. Quit
>> 3

Thank you for using my 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.

**Algorithmic design:**

a. Identify and list all of the user input and their data types.

- userChoice as double  (to read there options)
- principalAmount as double (beginning number)
- intersetRate as double (the rate of which number is changing)
- timeYear as double(number of time it changes in a year)
- compoundingPeriod as int( amount of time it compounds in a year)- choice 2

b. Identify and list all of the user output and their data types.

- InterestAccrued as double (amount of money from interest)
- totalAmount as double(total money with principals and interests)

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.

- SET double InterestAccrued = totalAmount- principalAmount
- SET double simple interest  = principal Amount(1 + intersetRate/100  * timeYear)
- SET double compound interest = principalAmount(1 + interestRate /100/ compoundPeriod) power of compoundPeriod * timeYear

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.

1. FUNCTION welcome()
    a. `Welcome to my Interest Calculation program!`

   END FUNCTION welcome()

2. FUNCTION readNow(string prompt) (Data Validation function for an integer)
    a. DECLARE option as double
    b. INPUT into option
    c. DO WHILE
        i. IF userChoice  not positive
        ii. DISPLAY error message
        iii. RESET input stream
        iv. CLEAR buffer

v. INPUT into option

d. END WHILE
e. return option

END FUNCTION readNow()

3. FUNCTION displayMenu()
   a. 
```
Pick an option from below:
    1. Simple Interest
    2. Compound Interest
    3. Quit
```
END FUNCTION displayMenu()

4. FUNCTION readOption()
   a. CALL readNow function to read userChoice
   b. return userChoice

END FUNCTION readOption()

5. FUNCTION exeOption(double userChoice)
   a. CALL readInt function to read principalAmount
   b. CALL readInt function to read intersetRate
   c. CALL readInt function to red timeYear
   d. IF userChoice is 1 THEN
      i. totalAmount = calcAmount1(principalAmount, intersetRate, timeYear)
   e. ELSE IF userChoice is 2 THEN
      i. CALL readInt function to read int compoundPeriod
      ii. totalAmount = calcAmount2(principalAmount, intersetRate, timeYear, compoundYear)
      iii.
   f. ELSE
      i. return 0
   g. SET double InterestAccrued = totalAmount- principalAmount
   h. DISPLAY “Interest accruals: $:” and interestAccrued
   i. return totalAmount

END FUNCTION exeOption

6. FUNCTION simpleCal(double principalAmount, double intersetRate, double timeYear)
   a. returnprincipalAmount(1+ intersetRate/100) * timeYear)

END FUNCTION simpleCal()

7. FUNCTION compoundCal(double principalAmount, double intersetRate, double timeYear, double compoundingPeriod)

a. return principalAmount(1+ (intersetRate/100) / compoundPeriod) power of compoundPeriod * timeYear

END FUNCTION compoundCalCalories()

8. FUNCTION main()
    a. DECLARE userChoice as double
    b. DECLARE totalAmount as double
    c. CALL welcome()
    d. IF userChoice is 3
        i. Return 0;
    e. DO LOOP
        i. CALL displayMenu()
        ii. CALL readOption and get userChoice
        iii. CALL exeOption(userChoice) and get totalAmount
        iv. DISPLAY Total accrued amount: totalAmount
    f. END DO WHILE userChoice != 3:
9. DISPLAY Thank you message.
10. END FUNCTION main()

## 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<br>*statement*<br>*statement*<br>END IF | `IF num_dogs > 10 THEN`<br>`  DISPLAY "That is a lot of dogs!"`<br>`END IF` |
| Use a dual alternative conditional | IF *condition* THEN<br>*statement*<br>*statement* | `IF num_dogs > 10 THEN`<br>`  DISPLAY "You have more than 10 dogs!"` |

| | | |
|---|---|---|
| | ELSE<br>  *statement*<br>  *statement*<br>END IF | ELSE<br>  DISPLAY "You have ten or fewer dogs!"<br>END IF |
| Use a switch/case statement | SELECT *variable or expression*<br>  CASE *value_1:*<br>  *statement*<br>  *statement*<br>  CASE *value_2:*<br>  *statement*<br>  *statement*<br>  CASE *value_2:*<br>  *statement*<br>  *statement*<br>  DEFAULT:<br>  *statement*<br>  *statement*<br>END SELECT | SELECT num_dogs<br>  CASE 0: DISPLAY "No dogs!"<br>  CASE 1: DISPLAY "One dog.."<br>  CASE 2: DISPLAY "Two dogs.."<br>  CASE 3: DISPLAY "Three dogs.."<br>  DEFAULT: DISPLAY "Lots of dogs!"<br>END SELECT |
| **Loops** | | |
| Loop while a condition is true - the loop body will execute 0 or more times. | WHILE *condition*<br>  *statement*<br>  *statement*<br>END WHILE | SET num_dogs = 1<br>WHILE num_dogs < 10<br>  DISPLAY num_dogs, " dogs!"<br>  SET num_dogs = num_dogs + 1<br>END WHILE |
| Loop while a condition is true - the loop body will execute 1 or more times. | DO<br>  *statement*<br>  *statement*<br>WHILE *condition* | SET num_dogs = 1<br>DO<br>  DISPLAY num_dogs, " dogs!"<br>  SET num_dogs = num_dogs + 1<br>WHILE num_dogs < 10 |
| Loop a specific number of times. | FOR *counter* = *start* TO *end*<br>  *statement*<br>  *statement*<br>END FOR | FOR count = 1 TO 10<br>  DISPLAY num_dogs, " dogs!"<br>END FOR |
| **Functions** | | |
| Create a function | FUNCTION *return_type name (parameters)*<br>  *statement*<br>  *statement*<br>END FUNCTION | FUNCTION Integer add(Integer num1, Integer num2)<br>  DECLARE Integer sum<br>  SET sum = num1 + num2<br>  RETURN sum<br>END FUNCTION |
| Call a function | CALL *function_name* | CALL add(2, 3) |
| Return data from a function | RETURN *value* | RETURN 2 + 3 |