# CIS11 Course Project Part 1: Documenting the Project

By Senelli Jinadasa & Jeanine Rodriguez

**Introduction**

* 1. **Purpose**

This document details the functionality and requirements of the Test Score Calculator.

* 1. **Intended Audience and Users**

The primary audience/user are students and teachers

* 1. **Product Scope**

What is the intention of this program?

The Test Score Calculator program is intended to replace most existing grading / test calculator programs with certain features that are mentioned in this document. This program provides an easy and new way to check out what are the highest, lowest, and average test scores one may have.

* 1. **Reference**

**Source Documents for the Program Requirements and Specification**

Reference Project requirements and LC-3 specifications.

1. CIS Project Participants

<https://docs.google.com/document/d/1tqD_CCqGzy7LF6YSuBc5MQEHwZunbGpVRQSeAJpaH_A/edit>

1. Project Criteria

[CIS11 Course Project Part 1 FINAL-1.docx](https://studentrcc-my.sharepoint.com/:w:/g/personal/jrodriguez779_student_rccd_edu/ESp_OZ0fNYBIkzNGXY1tiRsBnyImyUp1kHthT52T8_tF0A?e=fjQuMA)

1. CIS Project Flowchart

<https://docs.google.com/drawings/d/13GNKd_Vy_4y5lmoc0Y6FGApRKYhwVwrizci9Vy6NeeI/edit>

1. CIS11 Project documentation
2. GitHub repository (README.MD)

**2. Overall Description**

**2.1 Product Perspective**

Primary program objectives

The Test Score Calculator provides:

* A user interface (console) to take input (five test scores) from the user
* The grade for each of the test scores
* The minimum and the maximum test score and display them onto the console for the user to see
* Calculated average of the test scores and display them onto the console for the user to see
  1. **Product Functions**

**The overall description of functionality:**

Highlight the program functionality: Identify tasks and subtasks of the program in summary.

1. The program gives the user (instructor or student) a user interface with clear instructions on how to input each test score
2. After a test score is input into the program by the user, the program will display the grade for the test score
3. After all test scores have been input in by the user, the program will find the minimum test score, the maximum test score, and calculate the average test score
4. Display the minimum, maximum, and average test score onto the console for the user to see

**Technical functionality**

A configurable toolkit of functions including:

What are the technical functions of the program? Subroutines and operations.

* + Ability to take in user input (ASCII values) and convert them to hexadecimal values in order to do the calculations required (find minimum, maximum, and average) by the program
  + Ability to create a data container to store the test scores
  + Ability to find the minimum test score in the test scores data container (by looping through the data container)
  + Ability to find the maximum test score in the test scores data container (by looping through the data container)
  + Ability to calculate the average by adding all the values in the test scores data container using pointers and calling another subroutine to divide the result by the number of items in the data container
  + Ability to convert the minimum, the maximum, and the average test score values which are in hexadecimal to ASCII values, and display them onto the console for the user to view
  1. **User Classes and Characteristics**

**Who** **are involved in this development process? Include business and technical personnel and their tasks.**

Jeanine Rodriguez and Senelli Jinadasa were both involved in the development process of the Test Score Calculator program.

**Jeanine Rodriguez:**

Responsible for creating the flowchart during the beginning of the development.

Responsible for updating their flowchart with the development process

Also responsible for documenting the development process of the program

Updating additional information to the documentation document.

Had a part in running and testing the program to ensure the program is running smoothly and check for any problems that may occur during the testing phase.

**Senelli Jinadasa:**

Responsible for making the pseudocode in the beginning of development

Responsible for keeping their pseudocode up to date with the program

Updating their teammate on their progress

Also responsible for creating and developing the test score calculator program.

Was also involved in testing and running the program

Checking for any errors the program may have and fixing it.

Also helped with the documentation

* 1. **Operating Environment**

What type of system will the application be operated on? Operating system? System types? Development platform?

Test score Calculator is developed for use on Windows and a Unix system

Test Score Calculator operates on the LC-3 Simulator application

Optional: could also operate on web-based browsers and the internet

* 1. **Design and Implementation Constraints**

Note any constraints or limitation to the application.

User limitations include not being able to type more than three digits per score. Another limitation is that the program does not restart and must manually reset it in the simulator. As for the developer limitations, the program will only display the min, max, and average and not the rest of the test scores.

* 1. **Assumptions and Dependencies**

Note any dependencies

It is assumed that the user data will need to be reinitialized after its completion. Assuming the user is familiar with using the LC3 program or a web-based Simulator. User should already be familiar with mouse and keyboard.

Since the Test Score Calculator program is not a web-based application, there is no need for the internet. It is assumed that users already possess the LC-3 application.

If the user chooses to use an Online Simulator or does not have an application, then it is assumed that the user will posse’s decent internet to access the application and/or the web.

***3*. External Interface Requirements**

* 1. **User Interfaces**

How will the user interface with your program? Menus? Access prompt? Links? Icons?

The user must enter five scores into the prompt by using the console program. The User can input data by using the numbers on the keyboard and press enter when necessary. The program provides an opening prompt from which the user can insert their scores and see their results.

* 1. **Hardware Interfaces**

Specify hardware interface – computer types? Terminal types?

The computer hardware interface requires a keyboard so that the user can type their scores into the program. Another hardware interface requirement would be a mouse and a monitor because it allows you to access and run the program with one click.

* 1. **Software Interfaces**

Specify additional software interface – if any. What type of software will the application require to run?

The software that this program requires to run is the LC-3 Simulator. Depending on the system the user is on, there is a windows version 3.01 and a Unix simulator/Linux.

* 1. **Communications Interface**

Does your application require web, Internet, or network connectivity? If so, which browser? What type of network connection?

This application does not require web, internet, nor network connectivity. This application runs offline on the LC-3 simulator. Unless the user chooses to use an online simulator, then you would use the internet. Browsers such as Chrome or Microsoft edge would be fine.

**4. Detailed Description of Functional requirements**

**4.1     Type of Requirement (summarize from Section 2.2)**

**What are the functions? Their purposes? Inputs? Outputs? Data? Where is the data stored (internal or external to the application)?**

**Prompt Requirement**

**Purpose:** Provides a prompt “Enter a Test Score:” five times.  
**Inputs:** Inputs are through the keyboard.

**Processing: The** input is validated each time the user submits a test score.

**Outputs:** The correct input will result in the program calculating and sorting the users scores. If the input is incorrect then an error message will show up.

**Data:** User database

**Analysis Requirement**

**Purpose:** The 5 test scores are submitted to perform an analysis and sort them by minimum, maximum, average, and letter grades.  
**Inputs:** Input will be the test scores used by what the user wants to use for the analysis.

**Processing: Depending** on the test scores, the program will calculate and analyze the data by using subroutines, converting ASCII characters to hexadecimals.

If an invalid input is used, there will be error messages handles using LC-3 exceptions in the LC-3 programs (LC3 Editor & LC3 Simulator).

**Outputs: The** output will be the test scores sorted by minimum, maximum, and average results while showing letter grade as well. These results will be displayed on the LC-3 Console.

**Data:** Program Database

**4.2 Performance requirements  
 What is the expected performance level of the program?**

**4.2.1** The application should be simple and quick for users of the Test Score Calculator to use.

**4.2.2** Since the application will be displaying the test scores from the user’s input, the program should display the minimum, maximum, and average with a letter grade.

**4.2.3** The database must have the capacity to hold numbers such as 1 -100 for users in the future.

**4.2.4** Error handling should be implemented, and the application should be able to handle all errors such as numbers over 100, symbols and letters.

**4.3 Flow Chart and Pseudocode.**

**Pseudocode:**

Load ASCII conversion to a register

Do Two’s complement of the ASCII conversion

Load count which is initialized to 5 onto a register

Prompt User for input

For loop for getting user input and storing it into an array

Get user input

Convert ASCII characters to Hexadecimal values

Call HUNDREDS subroutine to convert value to a hundreds value

Load value from HUNDR100 to a register

Get user input

Convert ASCII characters to Hexadecimal values

Call TENS subroutine to convert value to a tens value

Load value from TEN10 to another register

Get user input (R0 (Register 0))

Clear an unused register (for example R5)

Add R0 (new user input) and HUNDR100 and move it to a register (R1)

Add TENS10 and value in R1 and move it to R5

Point to Value in TEST\_SCORE array using the count

Store value in R5 (all values added) into the TEST\_SCORE array at given

Decrement count Until count is 0

If count is 0, go to CONT

CONT

Load stack (BASE into memory address of the stack) into R6

Call AVERAGE function

Load AVG\_TEST to R0

Add zero to R0

Call PUSH subroutine

Clear R0

Call MAX\_VALUE function

Load MAXIMUM to R0

Add zero to R0

Call PUSH subroutine

Clear R0

Call MIN\_VALUE function

Load MINIMUM to R0

Add zero to R0

Call PUSH subroutine

Clear R0

UNSTACK

Call ISEMPTY subroutine to see if stack is empty

If ISEMPTY returns positive (stack is empty):

Go to UNSTACK\_END

Call POP subroutine

Add 30 to ASCII convert value

Display value to console

Branch UNSTACK unconditionally until stack is empty

UNSTACK\_END

Halt program

Subroutine HUNDRED

Clear two registers (R2 and R6)

Set Counter1 to 10 (loop counter)

Set Counter2 to 10 (loop counter)

For loop MLOOP:

Add first user input (R0) to one of the cleared registers (R6)

Decrement counter1 until counter is 0

Branch to MLOOP if counter1 is still positive

If counter1 is positive go back to MLOOP

If counter1 is zero or negative continue to MLOOP2

For loop MLOOP2:

Add another cleared register (R2) and second user input (R0)

Decrement counter2 until count is 0

If counter2 is positive go back to MLOOP2

If counter2 is zero or negative continue to ENDLOOP

ENDLOOP

Store the value in R2 in a temporary memory location HUNDR100

Return

Subroutine TENS

Clear a register (R2)

Set Counter1 to 10 (loop counter)

For loop MLOOP1:

Add first user input (R0) to the cleared registers (R2)

Decrement counter1 until count is 0

Branch to MLOOP1 if counter1 is still positive

If counter1 is positive go back to MLOOP

If counter1 is zero or negative continue to MLOOP2

ENDLOOP2

Store the value in R2 in a temporary memory location TEN10

Return

Subroutine MIN\_VALUE:

Clear register R1 and R2

Point to first value in TEST\_SCORES array using R1

Store R1 in MIN temporary memory location

Set count1 to 4 (loop counter)

For loop MINIMUM:

Increment the value inside of R2

Use the value of R2 to point to an element in TEST\_SCORES array

Do two’s complement of the value above (element in array)

Load MIN value to a register (R1)

Add two’s complement of R2 and MIN (R1) value

If result is negative:

Store R1 in MIN temporary location

Decrement count1

If count1 is still positive go back to MINIMUM

If count1 is negative go to ENDMIN

If result is zero or positive:

Store R2 in MIN temporary location

Decrement count1

If count1 is still positive go back to MINIMUM

If count1 is negative go to ENDMIN

ENDMIN

Return

Subroutine MAX\_VALUE:

Clear registers R1 and R2

Point to first value in TEST\_SCORES array using R1

Store R1 in MAX temporary memory location

Set count1 to 4 (loop counter)

Clear register for

For loop MAXIMUM:

Increment the value inside of R2

Use the value of R2 to point to an element in TEST\_SCORES array

Do two’s complement of the value above (element in array)

Load MAX value to a register (R1)

Add two’s complement of R2 and MAX (R1) value

If result is negative:

Store R2 in MAX temporary location

Decrement count1

If count1 is still positive go back to MAXIMUM

If count1 is negative go to ENDMAX

If result is zero or positive:

Store R1 in MAX temporary location

Decrement count1

If count1 is still positive go back to MAXIMUM

If count1 is negative go to ENDMAX

ENDMAX

Return

Subroutine AVERAGE:

Clear registers R2, R3, R4

Count2 set to 5

For loop AVG:

Use the value of R2 to point to an element in TEST\_SCORES array

Add the value pointed to in array and move to register R4

Increment R2

Decrement Count2

If count2 is positive:

Go back to AVG loop

If result is zero or negative:

Store R4 in SUM\_TEST temporary location

Call DIV subroutine

Return

subroutine DIV:

Clear register R1

Clear register R4

While loop DIVLOOP:

Load SUM\_TEST to R4

Decrement R4 (SUM\_TEST) value by 5 (subtract 5 from R4)

If value in R4 is zero or positive:

Add one to R1 (increment R1 by 1)

Loop again (go back to DIVLOOP)

If value in R4 is negative:

Store R1 in AVG\_TEST temporary location

Go to END\_DIV

END\_DIV

Return

PUSH subroutine:

Push values passed in onto the stack

Return

POP subroutine:

POP values passed in onto the stack

Return

ISEMPTY

Checks to see if stack empty

Return a value of 1 if stack empty

**Flow Chart:**

Diagram

Description automatically generated