# CIS11 Course Project Part 1: Documenting the Project

**Introduction**

* 1. **Purpose**

**This document explains how to use our Test Score Calculator program as well as how it works. This program was created to put our LC-3 skills to the test and this document gives us a chance to communicate how our code works.**

* 1. **Intended Audience and Users**
* **Teachers/Students who need to quickly get the average of 5 test scores**
* **Professor Nguyen**
  1. **Product Scope**

What is the intention of this program?

**The Test Score Calculator’s purpose is to take 5 test scores and find the highest score, the lowest score, and the average of the 5 scores. The program outputs all three values and reports a final grade considering all scores.**

* 1. **Reference**

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

Reference Project requirements and LC-3 specifications.

[**https://www.youtube.com/channel/UC9XmHOtPAEZofFqVG\_1C10w/videos**](https://www.youtube.com/channel/UC9XmHOtPAEZofFqVG_1C10w/videos) :)

**2. Overall Description**

**2.1 Product Perspective**

Primary program objectives

**This program should be used to quickly calculate one’s overall grade.**

* 1. **Product Functions**

**The overall description of functionality:**

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

**This program:**

* **Asks the user for an integer input between 0-100**
* **converts the characters into a number the computer can perform operations on**
* **pushes the number onto a stack**
* **fetches the 5 numbers from the stack and compares them to find the highest and the lowest**
* **calculates the average by adding all 5 numbers and dividing them by 5**
* **converts the max, min, and average back to characters to be outputted onto the console** 
  1. **Operating Environment**

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

**This program can be implemented on any system that can run an LC-3 simulator. It can be assembled and ran on the web using** [**https://wchargin.com/lc3web/**](https://wchargin.com/lc3web/)

* 1. **Design and Implementation Constraints**

Note any constraints or limitation to the application.

**At one point the program got too big for Load instructions to find the data being asked for. This required the developers to find more optimized and clean solutions**

***3*. External Interface Requirements**

* 1. **User Interfaces**

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

**The user will only be prompted to input the numbers on the console log. The program goes to the next lines once a full number is inputted, so pressing enter is unnecessary.**

**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)?**

INPUT: Subroutine that is called 5 times to get 5 user test scores. The test scores are all converted from ASCII format into usable numbers and pushed onto the stack in the subroutine. It does not return data in the registers to be used in the main program.

MULT: R2 equals x and R5 equals y. The subroutine finds the product of x and y by adding x to R1 y times by decrementing R5, checking if it is less than zero, and if not, adding x to R1, then looping to the decrement instruction. If it is less than zero, the loop is exited and R1 equals the product.

FINDMIN: Gets the pointer to the top of the stack of test scores and checks all 5 of them (using the pointer) to find the lowest score. The subroutine sets the top of the stack as the MIN and subtracts the next value from the MIN value. If the result from this subtraction is positive, we know that the value being tested is smaller than MIN, making it the new MIN. This process repeats until all values are checked.

FINDMAX: Starts at the top of the stack and checks all 5 scores. The subroutine sets the top of the stack as the MAX and subtracts the next value from MAX. If the result is negative, the tested value must be larger, making it the new MAX. The process is repeated until all scores are tested.

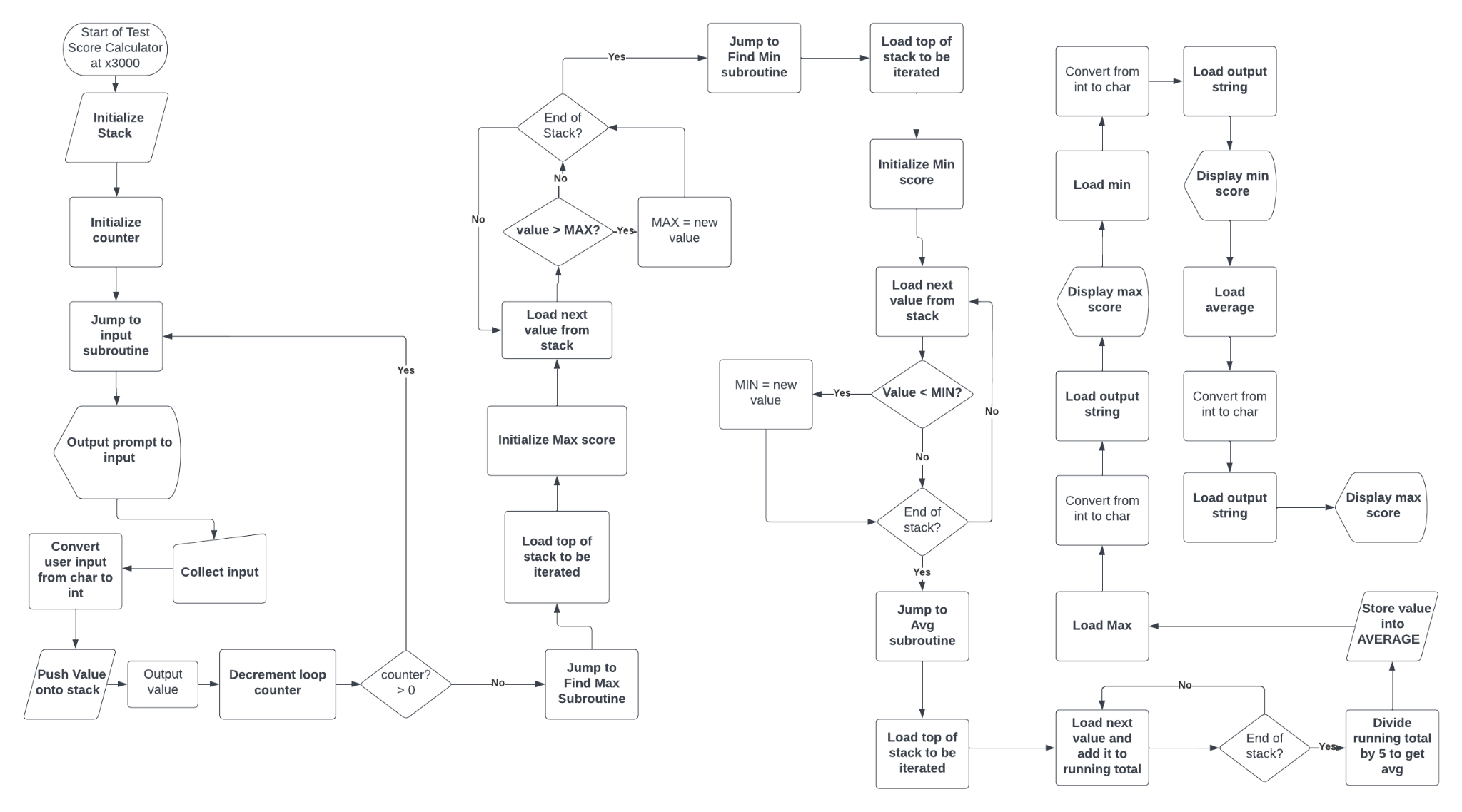
OUTPUT: This subroutine prints text stating what test score is currently being printed and the test score being analyzed, along with a letter grade for it. The arguments are the former two information stated, stored in R0 as a string and R5 as a number. R0 is printed and the hundreds, tens, and ones places are converted into separate digits with the DIV subroutine, then ASCII offsetted, and then printed in the correct order. The hundreds and tens place are then checked to see what grade the test score should receive, ranging from A to B, C, D, and F, after which that, along with a new line, is printed as well.

AVG: Points to the top of the stack and adds the value its pointing to R5. It then continually decrements the pointer to point to the next value and adds to R5 until all values are added. Then, it divides this number by a constant 5 and puts the result in R4, where it is stored into the label AVERAGE in the main program.

DIV: R5 equals the dividend and R2 equals the divisor. This subroutine finds the quotient and remainder of the dividend and divisor. This is done by subtracting R2 from R5, checking to see if the number is negative, and if not, incrementing the quotient, then checking to see if R5 equals zero, and if not, looping to the subtraction instruction. If R5 is negative before incrementing the quotient, then the loop breaks and R2 is added back to R5 to fix the remainder, and R5 then equals the remainder, R4 equals the quotient, and the subroutine returns. If R5 equals zero after the increment instruction, the same thing happens, except that the remainder is not fixed because this means the dividend was divided perfectly (no remainder).

MPROMPT: Due to the distance from the beginning of the program to the data, this subroutine was created to easily access the main prompt string and print it without any problems.

**4.3 Flow Chart and Pseudocode.**



|  |
| --- |
|  |  |