CIS11 Course Project Part 1: Documenting the Project

LC – 3 Grade Calculator Documentation Guide – 2022 (Immanuel Morris/Robert Hinojo)

Introduction

1.1 Purpose

The Purpose of this program is to successfully have a user enter a maximum of 5 test scores, and have the result displayed on console. The result will consist of a decimal percentage number entered, and the program will output a letter grade ranging from A-F, along with a necessary average in-between.

1.2 Intended Audience and Users

This program is intended for students and Instructors in an educational setting or environment.

1.3 Product Scope

The Test Score Calculator program is intended to produce a trio of maximum, minimum, and average test scores for the user to see. Extended features are explained further in this document, but in short, this program will ease the process of filtering through a grading process for an instructor to have accurate information to reliably record. In addition, students will be able to assess their current status based off of the recorded scores.

Reference

Companion Application Requirements Documents (If applicable)

1) Pseudo code (doc or txt)
2) Diagram/flowchart (pdf/jpg/doc)
3) Git-Hub developer link
4) Flow-Chart Diagram

2. Overall Description

2.1 Product Perspective

This Products Provides the following:

Fundamental and logical workflow for the following uses

- Taking user input of a set decimal (whole number). This will be internalized as a percentage
- Running through a series of customized functions, loops, and arrays, for an algorithmic process
- Display results based upon input, incorporating mathematical processes to determine averages, maximum, and minimum values.

2.2 Product Functions

The overall description of functionality:

1. Enables person(s), business, organizations, or entity's the option of accelerating the grading process on a digital format, versus manual input. From an instructor point of view, this will allow one to gauge where a student is relative to the course status.
2. Internally tracks score(s) for all students so that further assessment of future goals can be accommodated.
3. Allows for adjustment of a particular score to be adjusted, and overwritten if needed.
4. As previously stated, this program will allow faculty, and staff the opportunity to log and record scores utilizing customized software tailored specifically for percentage conversion to grade letter.
5. Program consists of customized internal functions that categorizes different processes for conversion to grade letters
6. Program will display easy to read result and conversion consisting of a maximum result of 5 test scores. These results will additionally benefit students for status in where they rank in their academics, and provide instructors, teachers, faculty, and staff, an opportunity to compare grade statistics against similar courses.
7. Program will not display any additional information outside of basic letter grade ranging from A-F in any order determined by the internal algorithm.

Technical functionality

This LC-3 program is comprised of custom subroutines and functions including but not limited to:

- Structured loops that will decrement based on the highest input value.
- Customized ASCII conversion translated to decimal value(s).
- Stored, and reused registers for memory manipulation and management
- Pointers for memory control and allocation.
- Additional subroutines categorized based off of (decimal) percentage input to direct the program in which grade to display.

2.3 **User Classes and Characteristics**

Project Department Head (Professor Nguyen):
Responsible for overall publishing and delivery of program.
Typically has the final say-so over what works in the program and what doesn't.
Program WILL NOT be published should the department head not approve or program contains missing parameters.
Steps through program line by line to confirm the logical necessary steps are implemented without memory error or taxing processing.
Supports developers on all levels for implementation guidelines, directions, and architecture.

Documentation Developer (Immanuel Morris):
Responsible for research that embodies the details that are contained within the program.
Tasked with communicating to the reader the fundamental functions of the program, and the basics of implementation along with computer specs.
Provides tips, tricks, shortcuts (if any) for users while running the program.
Displays thorough guidelines for running the program, and possible capabilities and limitations within a Mac or PC environment.

Software Developer(s) (Robert Hinojo/Immanuel Morris):
Responsible for the actual code written to produce and create the program.
This is typically a team of 2-3 developers with knowledge in basic data structure and constantly provide updates and revisions for code.
These persons are also tasked with debugging issues, and general oversight of the program.
They work directly under the Project department head.

Pseudocode/Flowchart Designer (Robert Hinojo):
Approaches from a blueprint perspective of what the overall program needs to do, prior to the actual program being created.
This person is also typically in the Software Development department, and must have a clear view and understanding of the idea of the program.
The designer is well versed in connecting the thoughts of the program in a node – constructed manner, and can convey the overall goal of the program in a diagram structure to give a visual representation of what the objective is.

2.4 Operating Environment

This Program is exclusively developed and implemented through the use of LC -3 (Little Computer 3)
LC-3 is developed for use on a PC environment with a Windows or Linux operating system.
The qualified System requirements include:
Windows 7- 64 bit
Windows 10 – 64 – bit (32 bit works also)
Minimum of 5GB of disk space for disk cache and temp files.

Mac OS environment is optional, but **MUST** be implemented using Java
*Java update in the works.

Other necessities include standard monitor, mouse, and keyboard
Preferably 16-32 gigs of ram to optimize memory performance

2.5 Design and Implementation Constraints

As previously stated, to run on Mac OS, Java is required.

LC-3 is primarily built to run on a Linux and Windows environment.

This program can also be run online, through various simulators. They can be found here:
Github:
https://wchargin.github.io/lc3web/

LC-3 Tutor:
http://lc3tutor.org/ (Links to an external site.)

Instructions on how to use a simulator:
http://pages.cs.wisc.edu/~markhill/cs252/Fall2009/includes/lc3editguide.html

2.6 Assumptions and Dependencies

Since LC-3 can be installed and run on a local machine, (PC/Windows/Linux environment) there is no need for a true dependency if the user meets this requirement. Internet access may not be necessary.

On the contrary, if the user does not have access to a Windows or Linux environment, they may access a simulator online. This is to assume that the user has basic knowledge of internet accessibility, and will be required to do so barring that is the necessary route they choose to take.

3. External Interface Requirements

3.1 User Interfaces

The user is required to interact with the program using a mouse (to click) and keyboard (to press keys) as necessary. It is an absolute requirement that the user has a monitor, as the program requires input on a visual console prior to execution. The console, is only displayed on a simulator which exclusively visible on a monitor. The simulator requires the loading and running of a ".obj" file, therefore the user must use dropdown menus, and save and load options to actually run the program.

3.2 Hardware Interfaces

No specific Hardware types outside of the outlined necessities in the previous "Operating Environment" section of this document.

3.3 Software Interfaces

Software exclusively runs on LC-3 software. No additional requirements necessary unless the user chooses to run an online simulator.

3.4 Communications Interface

This application does not exclusively require web, internet or network connectivity, but can be accessed if necessary. Should the user require an internet connection to run the simulation software, we suggest using Google Chrome, Safari, or Mozilla Firefox.

4. Detailed Description of Functional requirements

4.1 Type of Requirement (summarize from Section 2.2)

Processing:
The program will consist of a conjunction between organized subroutines, loops, and branches that will allocate information accordingly.
There is a subroutine (function) that will allow the program to streamline the input data to recognize which value is within a range from 90-99, which will produce the result of an "A"
Additional subsequent subroutines will follow as necessary with providing conversion from input ranges like 80-89, which will produce a "C" and so on.

Branch values will suffice as "if" and "if else" conditions depending on which step the program is on. For example, should there be a negative value processed, (which is expected to happen while converting 2's complements) the program

implements a Branch zero contingency to have information successfully moved into the appropriate register for memory processing.
Similar formulas happen throughout the program, until the end is reached, and the information the user inputs is calculated and accounted for.

Outputs:
Again, referencing the console, the user will see their result displayed on the console that is accompanied by the simulator. For this reason, it is imperative that you have an output device plugged into your computer, such as a monitor.
The correct corresponding grade is expected to be displayed on console. Should the user not input information within the instructed parameters, the program will display an error message to the user.

Data:
Referencing the Simulator:
As briefly mentioned, accompanying the console is the LC-3 simulator, which will display address information of necessary steps of the code. Should the user want a more in-depth visual representation of the program, the memory processing and register flow can be seen with the simulator. Additionally, the primary focus at this point of display is the console, therefore it is much easier to understand the necessary results by reviewing the output information.

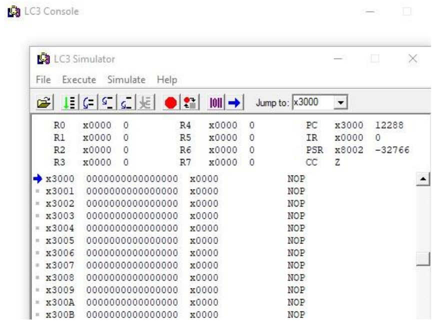
Purpose: To find the minimum, maximum, and average grade of at least 5 test scores. Additionally, display the accompanying letter grade to pair with the test score.

Inputs: Input is the sole responsibility of the user, and the information and data they so choose to input will allow the program to run its algorithm and display results.

Processing:
Based on the input from the user, the program will access a specific set of subroutines and branches to allocate the appropriate memory and registers to filter through. Additionally, there will be branch conditions set up as contingencies should the user not follow directions when prompted for input on the console.
The program will consist of a set of JSR's (Jump Subroutines) to jump to in regards to the range of input from the user. The program also has **Load** and **Store** practices available for memory to process through - as each register from R0-R7 is accounted for, cleared, and re-used. Also included, is a section set up for ASCII conversion should the decimal range not be enough for the program to filter through. This will be a series of register manipulations adding up a total sum of decimal value that will equate to an ASII Result. There will be an **Br** (Branch) section in place should the user enter out-of-bounds information, such as extreme negative values, or

values past 100. The program is not designed to display information that is not positive.

Outputs: As previously mentioned throughout various sections of this document, the output information will be exclusively displayed on the LC-3 console (this absolutely requires a monitor for visual display). To view information on the console, the user MUST import the accompanying ".obj" file, and load it into the simulator. Once that is set up, the user runs the program, and results are displayed onto the console. The simulator will allow the user to see the memory flow of the program, but the ultimate results and letter grade will only be displayed on the console.
See example below:



```
R0   x0000  0        R4   x0000  0        PC   x3000  12288
R1   x0000  0        R5   x0000  0        IR   x0000  0
R2   x0000  0        R6   x0000  0        PSR  x8002  -32766
R3   x0000  0        R7   x0000  0        CC   Z
x3000  0000000000000000  x0000          NOP
x3001  0000000000000000  x0000          NOP
x3002  0000000000000000  x0000          NOP
x3003  0000000000000000  x0000          NOP
x3004  0000000000000000  x0000          NOP
x3005  0000000000000000  x0000          NOP
x3006  0000000000000000  x0000          NOP
x3007  0000000000000000  x0000          NOP
x3008  0000000000000000  x0000          NOP
x3009  0000000000000000  x0000          NOP
x300A  0000000000000000  x0000          NOP
x300B  0000000000000000  x0000          NOP
```

4.2 Performance requirements

4.21 The application is expected to run on a local machine, but can be transferred to a Universal Serial Bus (USB) or a portable external drive. Additionally, the program can be run on the internet.

4.2.2 The expected response time of the program is within 1-2 seconds after recording input, and the user pressing "enter" on their keyboard. This program has minimal processing power, and is inexpensive time cost to the RAM.

(assuming the user meets the computer requirements)

4.2.3 The LC-3 program is absolutely scalable, and adjustments to the code can always be updated. This particular program holds the capacity for increased lines of code, functions, and algorithmic placement without disrupting the memory allocation of the OS.

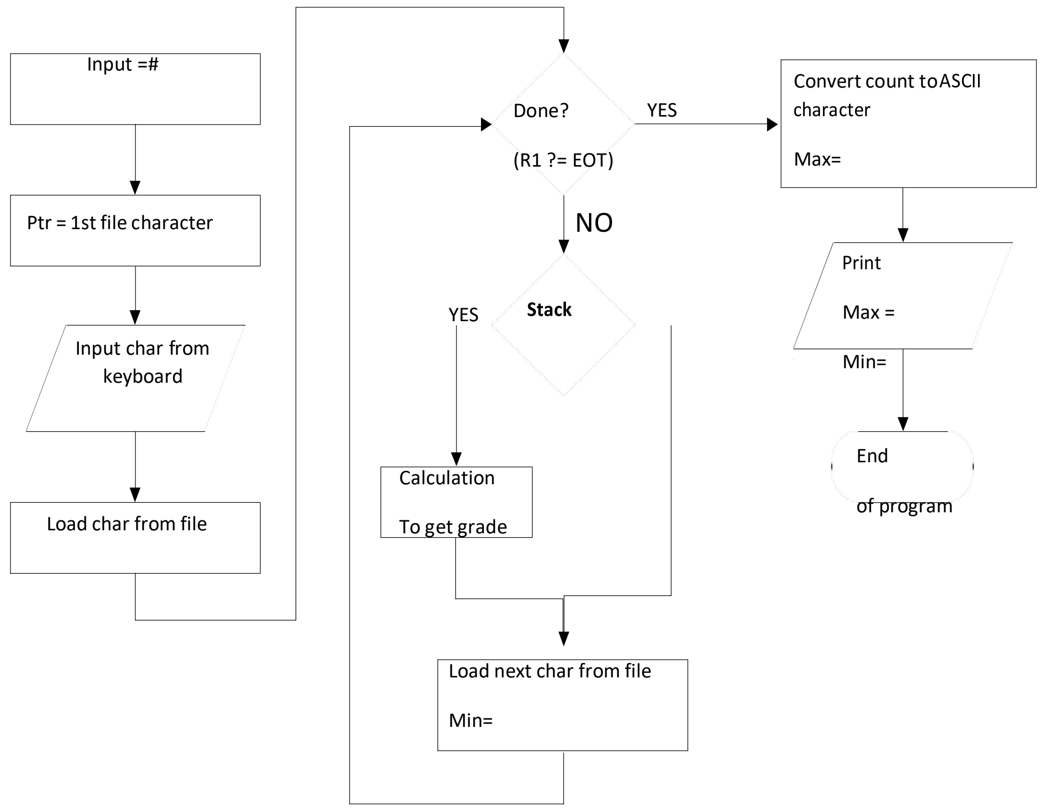
4.2.4 Contingencies for errors are in place should the user create input that is beyond the scope of the program.

4.3 Flow Chart and Pseudocode.

Flow Chart

Input: M[x3101] (Address)

Output: Print to display



This Page Intentionally left Blank