CIS-11 Project Documentation

Team Name: B. Awesome
Team Members:
Jason Ash
Ana Moreno
Luis Lopez

Project Name: Option B: Test Score Calculator
May 9, 2025

Advisor: Kasey Nguyen, PhD

Part I – Application Overview

This project is intended to fulfill the course requirements of CIS-11 Computer Architecture and Organization: Assembly at Moreno Valley College. There is not a company, commercial use, or business process, per se. At our fictitious company B. Awesome we are developing LC-3 software for course instructors to use to enter five integer test scores and display the minimum, maximum, average (arithmetic mean) and corresponding letter grade for each. Successful completion of this software will increase our company's reputation.

LC3edit (version 3.0.0.0) running on a compatible version of Microsoft Windows is required to assemble the source .asm file into the necessary .obj file and other associated files to run on the LC-3 Simulator. The LC-3 Simulator (version 3.01.0) also runs on Microsoft Windows which runs on an Intel-compatible x86 or AMD 64-bit microprocessor.

Objectives

Option B: Test Score Calculator

- User is prompted to enter five integer test scores
- The input is ASCII-converted to corresponding integers.
- Input validation: Test scores can't be less than 0 or greater than 100.
- Find the minimum, maximum, and average test scores as integers, and display the corresponding letter grade next to them.

Objective: Plan, document, and write a successful, functioning program in LC-3 assembly language that fulfills the stated selected project objectives.

Why are we doing this?

In addition to being required for the CIS-11 course, this project will increase the positive reputation of the programmers involved upon its completion. Many other aspects of the project, such as, flow charting, pseudocode, and this documentation will help us accomplish the objective stated above.

This program will simplify the workflow for course instructors entering five test grades since all they will have to do is enter them after the prompt, and the minimum, maximum, and average grade and corresponding letter grade will display. End-users will be satisfied that this tool is available to them and they do not have to do this work by hand. The LC-3 can also be piloted at educational institutions by utilizing the five test averaging program. The LC-3 is freeware, and the test averaging program is offered free of charge and will be open-source upon completion. Therefore, organizations will not incur any cost beyond an internet connection to download the LC-3 and this program to use it.

There is never a better time to work on this project because it is required to earn an A in this course. If work on this project were delayed too much, then our grade would be marked down or we would not receive credit for it at all. If we decided not to work on this project, we would miss out on the learning opportunity that it presents to tie together all of the concepts previously covered

in this course.

Stakeholders who will benefit from this project include end-users who will have a handy and reliable way to average five test scores. Also, the student team members will benefit from this project by the learning opportunity it presents and by earning the points towards their grade in the class.

While it might not be the best way to average five tests grades, it is a reliable and efficient way that beats doing this task by hand. Why use a complicated GUI when you can use a simple and efficient LC-3?

Business Process

This program will be a welcome addition to anyone still averaging five test scores by hand. Why spend precious time with the tedious process of averaging test scores by hand when an LC-3 program can do it for you? This is a welcome addition to the grading process used by educators. Averaging test scores is a common process used by many course instructors and teaching assistants throughout the world. Someday, this program could even be improved to average more test scores, additional assignments, and for the whole class instead of just one student at a time.

User Roles and Responsibilities

The users of this program could be instructors, teaching assistants, or students who want to average five test scores. As stated above, averaging test scores, determining the minimum and maximum test score, and their corresponding letter grade is almost a universal task carried out by teachers world wide and nearly since the beginning of civilization.

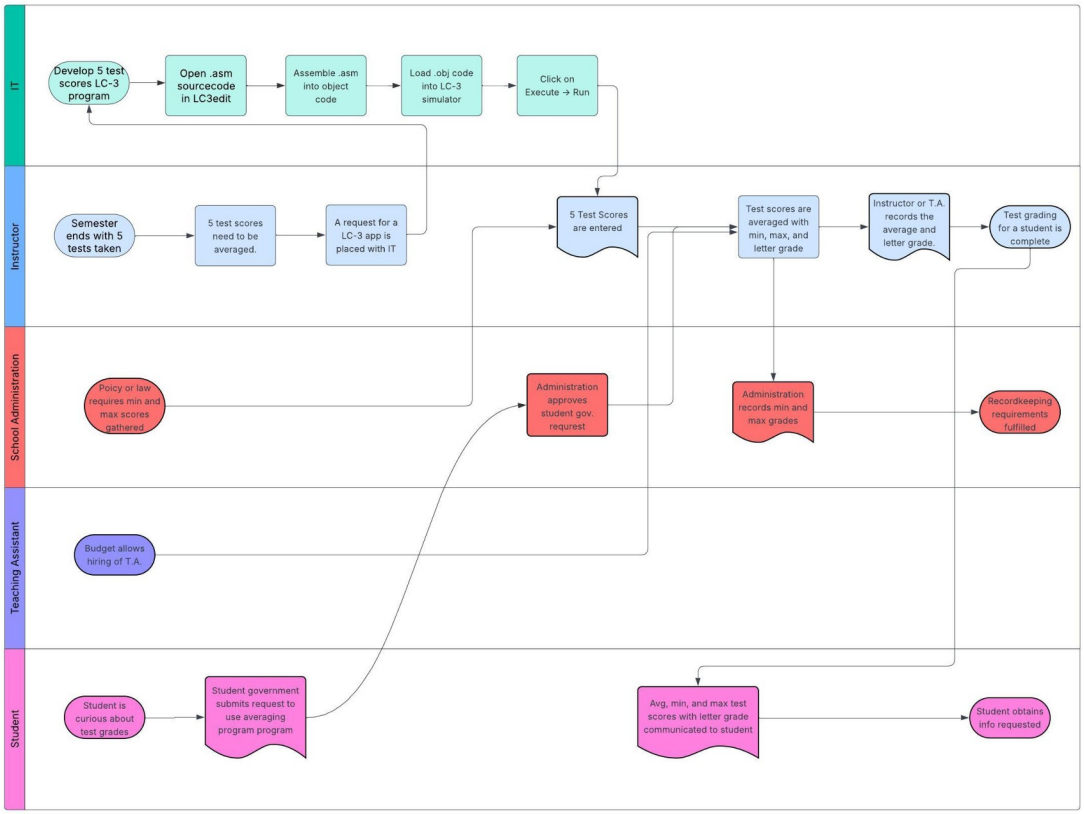
It might be necessary for an IT person or someone who is technically inclined and familiar with the LC-3 system to assemble the program and load its object file into the simulator. But once that is done and the program is ran, presto! Almost any member of the general public can follow the prompt and enter five test scores.

Since an IT person might not always be available, the tasks that involve the use of the LC-3 and the test averaging program are as follows:

- A reliable electric supply
- An Intel x86 or AMD x86-64 computer running Microsoft Windows with sufficient RAM, a keyboard, mouse, and monitor capable of reliably displaying the LC-3 simulator in the correct resolution, aspect ratio, and the appropriate number of onscreen colors.
- Download LC3edit and the simulator from our textbook authors' (Yale N. Patt and Sanjay J. Patel) Website at https://highered.mheducation.com/sites/0072467509/student_view0/lc-3_simulator.html
- The .asm source code file should be opened in LC3edit.exe and assembled by doing these steps:
  - File →Open, navigate to where the .asm file was saved or downloaded to, and double click on it.
  - Click on Translate → Assemble
- The user should double click on the simulate.exe program
- Click on File → Reinitialize Machine
- Click on File → Load Program and select the object file (.obj) created by the assembler.
- Click on Execute → run
- Make sure the console window is visible, click on it to give it focus, and enter the five test grades as prompted.

This program may be used whenever someone has five integer test scores (or any five equally weighted assignment scores in general) which could be one or more times per semester. It could also be ran to average a test set of five hypothetical test scores.

A hypothetical swimlane workflow diagram was made in Lucidchart for an organization, such as an educational institutions, that would need five test scores averaged with the minimum and maximum grade and corresponding letter grade displayed.



Production Rollout Considerations

The production rollout will depend on how team B. Awesome divides and delegates the tasks between us. Team B. Awesome will manage source code changes and revisions on GitHub. Discord will be used for discussions, and Zoom will be used for teleconferencing periodically as needed.

The data (variables and constants) contained within the program will be necessary for it to perform its function as a test averaging program that also displays the maximum and minimum test score and the corresponding letter grade of each. The test scores themselves will be entered by the user. The test data according to the requirements document will be the following five test scores: 52, 87, 96, 79, and 61. The test score that should display as the minimum is 52 along with the letter grade ‘F’. The maximum test score that should display is 96 with the letter grade ‘A’. The average test score should be 75 with the letter grade ‘C’ displayed next to it.

The expected data is five test scores between 0 and 100 each and LC-3 will have to be re-initialized, the object file reloaded, and the program reran every time the user wants to average another five test grades. (Unless we make a loop to make the program start over and a certain menu selection terminate the program.

The expected transaction volume is how many educators (or allowed students) use this five test score LC-3 program. Per program run, it is expected that the user will enter their complete list of five test scores to average and view the output displayed and record it in some fashion (probably by writing down the results since this program lacks functionality to save to secondary storage or print to a printer). The program will be ran at least once per semester for those semesters in which five test were taken by a student. Alternatively, if one test was taken by five students, and the educator wanted to know the average of five classmates, then that could also be a use for this program. It is expected that the former will be the primary use case. The program will have to be re-ran for every student in the class until all of their test scores have been averaged.

Terminology

Business or Technical Term	Definition
Addressing modes	Loading and storing data from RAM to registers and from registers to RAM, respectively, either directly, indirectly, or from an address or offset stored in a register or memory address.
Administrators	School principals and their delegated clerical staff that will use this LC-3 program to fulfill the regulatory requirements of recording the minimum and maximum test score for each student. This program does not have an administrator (i.e., superuser) and user mode. It functions the same for all users.
ASCII	The American Standard Code for Information Exchange is an eight-bit standard for representing characters and text input via a keyboard (Patt and Patel 47).
.asm file	An assembly language source code file.
Assembly language	A low-level programming language that usually has a 1:1 correspondence with instructions understood by the ISA of the computer.
B. Awesome	The team that programmed the LC-3 five test score averaging program and a name of a fictitious company in this document.
Bit	The most basic unit of information in a computer that is usually represented as on or off corresponding to the zeros and ones of binary

	logic.
Byte code	A file with the zeros and ones understood by a computer's ISA.
CIS-11 Computer Architecture and Organization: Assembly	A course at Moreno Valley College as part of the requirements to fulfill an Associate's Degree in Computer Science or Computer Programming or Certificate in Computer Programming.
Computer	Synchronous finite state machine for running programs (Patt and Patel 85)
Computer hardware	Microprocessor, motherboard, RAM, secondary storage, and input and output devices.
Console	The window in the LC-3 simulator that displays the output of the program and can receive input from the keyboard.
Data types	Representations in the ISA of different types of data it can handle, such as, integers, characters, boolean, and floating point data.
End-users/ user	Anyone who runs the five test score program.
Floating point data types	Fractional numbers with a value after the decimal point. The LC-3 does not support floating point data types.
GUI	Graphical user interface of most modern computers which the LC-3 console lacks since It is a 16-bit computer with about 65kb of memory.
Instruction set architecture (ISA)	"[…] the complete specification of the interface between programs that have been written and the underlying computer hardware that must carry out the work. […] The number of opcodes, data types, and addressing modes [are] specified by an ISA […]" (Patt and Patel 17).
Instructors, Teachers, and Teaching Assistants	Educators who might use the LC-3 program and record the results in their grade book.
Intel x86 or AMD x86-64 microprocessor	A 32 bit or 64 bit microprocessor with an Intel/AMD compatible ISA.
Initialization/Reinitialization	Resetting the LC-3's state to the default, factory reset state where all of the user program memory locations and registers store zero values by clicking on File → Reinitialize Machine.
IT person	Someone employed or educated in the field of information technology or a related field to provide technical support and internal technical

	solutions.
LC3edit and simulator	Programs made by the textbook authors Yale N. Patt and Sanjay J. Patel to teach computer architecture, organization, and assembly language.
Microprocessor	Logic gates made of silicon transistors and associated resistors, capacitors, and other components.
Microsoft Windows	An operating system developed, marketed, and sold by the Microsoft Corporation.
Moreno Valley College	A community college located in Moreno Valley, CA.
.obj file	A translated .asm file that is close to byte code and is opened in the simulator to run the program.
Opcode	A bit pattern that the ISA can decode to figure out what instruction it is being asked to do (Patt and Patel 68).
Programmer	Anyone with the technical skills necessary to systematically decompose a problem and express the implementation of the solution in a programming language that can be ran on a computer.
Program	A well planned sequence of instructions for a computer to carry out a task, calculation, or computation.
Programming language	"[… a] 'mechanical language' […] use[d] in specifying a sequence of instructions to a computer." (Patt and Patel 16).
RAM	Random access memory which is the primary storage of a computer and holds both the instructions of a program and the data the it is processing. Random access because any address can be accessed as easily as any other address.
Swimlane Diagram	"A swimlane diagram is a type of flowchart that delineates who does what in a process." ("What Is a Swimlane Diagram?")
Synchronous finite state machine	A finite state machine can only be in one state at a time, transitions from one state to another are defined, and all states are defined. Synchronous means that each state transition happens in lockstep with a clock mechanism (Patt and Patel).

Transistor	A metal-oxide semiconductor that can act as a switch depending on whether voltage is applied to it and they can be combined together to form logic gates (Patt and Patel 59-61).

Part II – Functional Requirements

The entry of one or two digits for a test score followed by the space bar or Enter key will terminate entry for that test score and move to the next test score. Entry of any three consecutive digits will also terminate entry for that test score, and the program will insert a newline character and automatically move to allowing entry the next test score. This will continue until the program has determined that five integer test scores have been entered. It will then compute the average test score, determine which test scores entered were the minimum and maximum of the sequence of five test scores. The program will finally display on the LC-3 console:

- The string “Minimum score:” followed by its numeric test score and letter grade on the same line.
- A newline character
- The string “Average test score:” followed by its numeric test score and letter grade on the same line.
- A newline character
- The string “Maximum score:” followed by its numeric test score and letter grade on the same line.

Statement of Functionality

In addition to the functional requirements listed above, this program will not be able to handle fractional test scores (i.e., test scores with a decimal value) because the LC-3 does not support floating point data types. The average will be a truncated integer without a remainder and without a decimal value. This means that no rounding will occur. For example, if the average test score would be calculated using a different method (by hand or with the assistance of a calculator) to be 89.9, the five test scores LC-3 program will just display 89, essentially chopping off the decimal part.

Although this application may have several distinct categories of users (educators, administrators, and students), there will not be different functional requirements by user category and it will function the same for all users. The different uses by user category is not expected to change the functioning of the five test score LC-3 program in any way. Educators are expected to have the highest volume of use on the application followed by administrators and students. Administrators may use the system to fulfill regulatory requirements since in the User Roles and Responsibilities swimlane diagram, they are required to gather data on the minimum and maximum test score for each student.

As previously stated, this LC-3 program does not have a way to store the test scores, their average, minimum, maximum, or corresponding letter grade or what student they belong to in secondary storage (a hard drive, floppy disk, solid state drive, USB drive, tape drive, etc). It also does not have the capability to print output to a printer. Therefore, the user will have to record results by writing them down, taking a screen capture on Microsoft Windows, or taking a picture of the program output with a camera (such as those that are standard on smartphones).

Scope

Phase I will be the Planning and Documentation phase in which this document will be the deliverable submitted by Sunday, 25 May, 2025 at 11:59pm. Another component of the Phase I was the "Team Project – Find Your Team" that while technically due on Sunday, 25 May, 2025 at 11:59pm must be completed much sooner so that teams can be formed to work on this LC-3 course project program. Team B. Awesome was formed on 9 May, 2025.

Phase II will be the successful completion of a five test score averaging program in LC-3 assembly language that fulfills the requirements on the rubric and in the Functional Requirements and Statement of Functionality above. Phase III will be setting up a GitHub account with a README.md and completing the self and team evaluation questionnaires. Both Phase II and III will be delivered by Thursday, 12 June 2025 by 11:59pm since that is when they are due, that is the end of the semester, and no late projects are accepted.

Performance

The lag between entering any test score and being allowed to enter another test score should be less than a second on the LC-3 simulator running on any Intel or AMD processor made in the last 15 to 20 years. The lag between inputting a digit and within a test score and having it echoed on the console and having entry move to the next digit or test score should likewise be a fraction of a second. The output of the minimum, average, and maximum test scores along with strings denoting each and the character of the corresponding letter grade should take no more than a second or two since while the LC-3 may have computational limitations due to being a simulator of a computer running on Microsoft Windows, the underlying processor is ultimately an Intel or AMD processor that is doing billions of clock cycles per second (GHz clock rate) and one or more instructions per clock cycle.

Program Function	**Description**
Detects termination of entry for a given test score	The user must enter at least one digit. Pressing the space bar or enter key after the first or second digit is entered will terminate entry of that test score and it will be considered complete. Likewise, as soon as three consecutive digits are entered for a test score, the program will go to the next line, and entry for another test score can begin anew.
Detection of a total of five test scores entered	The program will keep track of how many test scores have been entered via a counter, and after five have been entered, it will branch to processing and computational functions to calculate the average test score and determine which of the five test scores entered was the maximum or minimum.
Average Test Score	The average test score will be integer division with any fractional part or remainder that would normally be calculated on a calculator truncated.

Minimum and maximum test scores	The minimum test score will be the lowest test score entered by the user and the maximum will be the highest. In the event of a tie for minimum or maximum, only one of the tied test scores will be entered. For example, if two test scores are 75 and they are both the minimum, then the minimum test score output will be “Minimum score: 75 C”, and this will display just once.
Corresponding letter grade	The program will display the letter grade as a character separated from the numeric test score by a space. The letter grade displayed will be according to the standard grade scale: 59 or below = ‘F’, 60 to 69 = ‘D’, 70 to 79 = ‘C’, 80 to 89 = ‘B’, and 90 to 100 = ‘A’.
Fractional test scores	Not supported.
Fractional average test score	Not supported.
Rounding average test score	Not supported.
Writing of test scores, average, minimum, maximum and corresponding letter grade to secondary storage.	Not supported.
Printing of test scores, average, minimum, maximum and corresponding letter grade to a printer.	Not supported.

Usability

Since the five test scores average program should be fairly responsive (very little to no lag on input) and processing and output only taking a second or two at the most, performance as it relates to usability should not be an issue. It or a technically-inclined user may be needed to assemble the LC-3 source code into object code and to load the generated object file into the simulator. Other than that, anyone with general knowledge of how to use a keyboard to enter simple data (such as test scores into a computer) and able to read the output and record it in some other medium should be able to use this program.

Documenting Requests for Enhancements

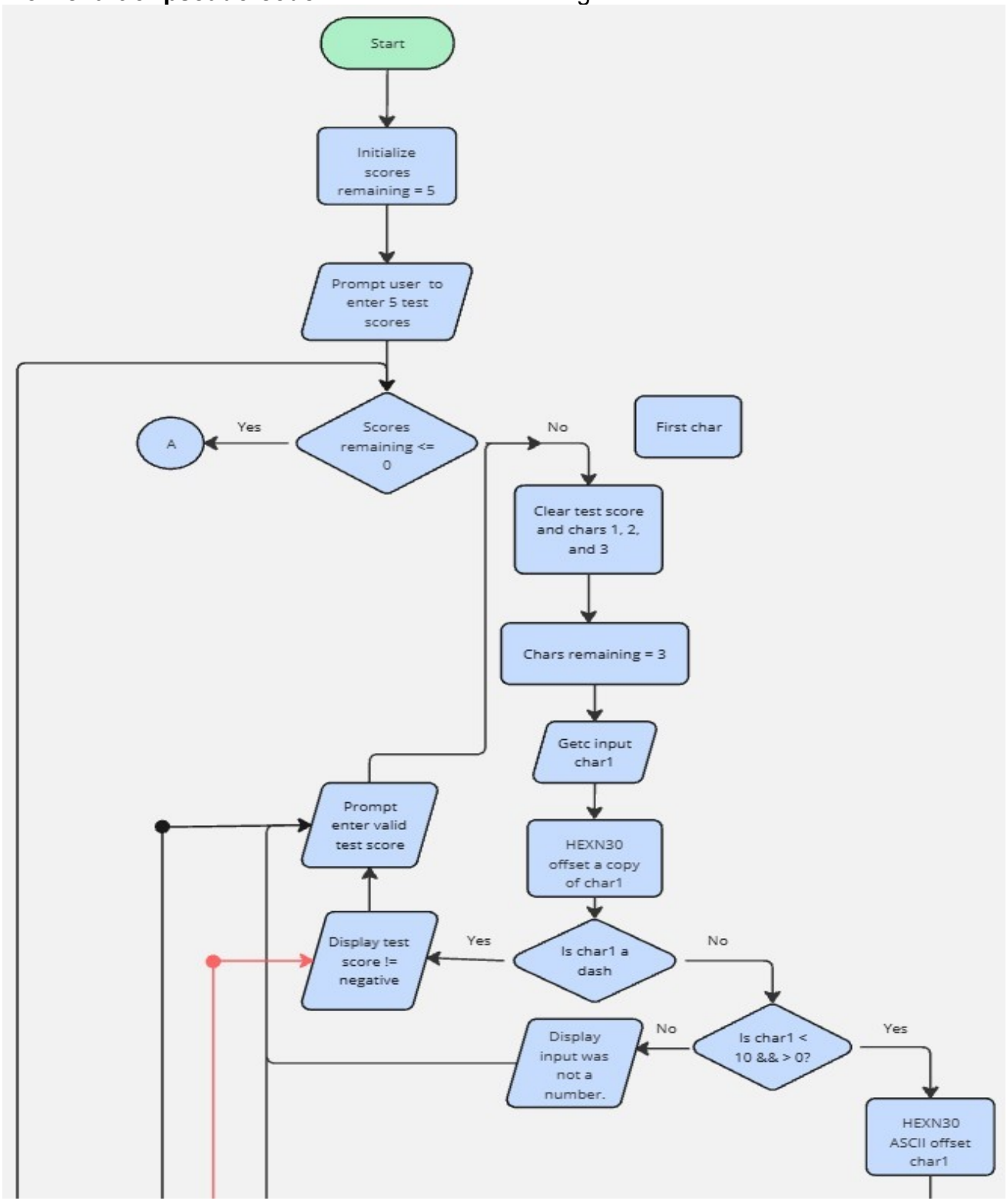
Date	Enhancement	Requested by	Notes	Priority	Release No/ Status
5/9/25	Sort and display test grades in descending order to delineate which is the max and min.	Jason Ash	This might require a bubble sort and be beyond the scope of this project.	Low	Not yet started
5/9/25	Bubble short to display test grades in ascending order.	Jason Ash	This might beyond the scope of this project and is almost essentially doing project A as a sub-project for B.	Low	Not yet started
5/9/25	Allow for backspace while entering characters	Jason Ash	This would be nice from a user interface perspective, but will it take too much time to implement and is it expected? The LC-3 may not be capable of backspacing an entry since it only displays a box character in the place of a backspace instead of deleting the character to the left as expected by pressing the backspace key.	Low/ not possible	Not yet started
5/9/25	Invalid char function for both second and third digits without reentry of previous chars?	Jason Ash	This would be nice because if the user has a typo on the second or third digits, then the previous digit will not have to be reentered. This might not make since, though, because if an invalid character is entered for the second or third digit, it would be better for the user to just reenter that test score.	Low	Not yet started
5/11/25	Average more than five test scores	Jason Ash	This is probably beyond the scope of this project.	Low	Not yet started
5/11/25	Average other assignments in additions to tests.	Jason Ash	This is probably beyond the scope of this project.	Low	Not yet started
5/11/25	Averaging test scores for all of the students in a class instead of just for one student at a time.	Jason Ash	This is probably beyond the scope of this project.	Low	Not yet started
5/11/25	Combining the multiplication functions for *10 and *100 into one function (if possible)	Jason Ash	Since both functions do multiplication, is there a clever way to just have one multiplication function?	Medium	Not yet started
5/11/25	Combining the hex negative 30 ASCII offset for the first second and third characters entered into one function.	Jason Ash	Since the hex ASCII offset to convert them to numeric digits is the same for all three characters, maybe one function can handle this for all of them.	Medium/Low	Not yet started.
5/13/25	Test averaging program correctly identify test scores over 100 instead of thinking the max is 199.	Jason Ash	Test scores should be from 0 to 100. Our program currently only displays an error if the test score is over 199.	High/critical	In progress
5/13/25	Test score program does not stop entry of test scores after five are entered.	Jason Ash	The test score program should only allow the user to enter five test scores and then do the processing and output according to its functional requirements.	High/critical	In progress
5/13/25	Fix multiplication subroutine(s) because if the test score is two or three digits, then the hex number placed on the	Jason Ash	The test score program needs to put the correct hexadecimal equivalent of a test score entered on the stack for later processing.	High/critical	In progress

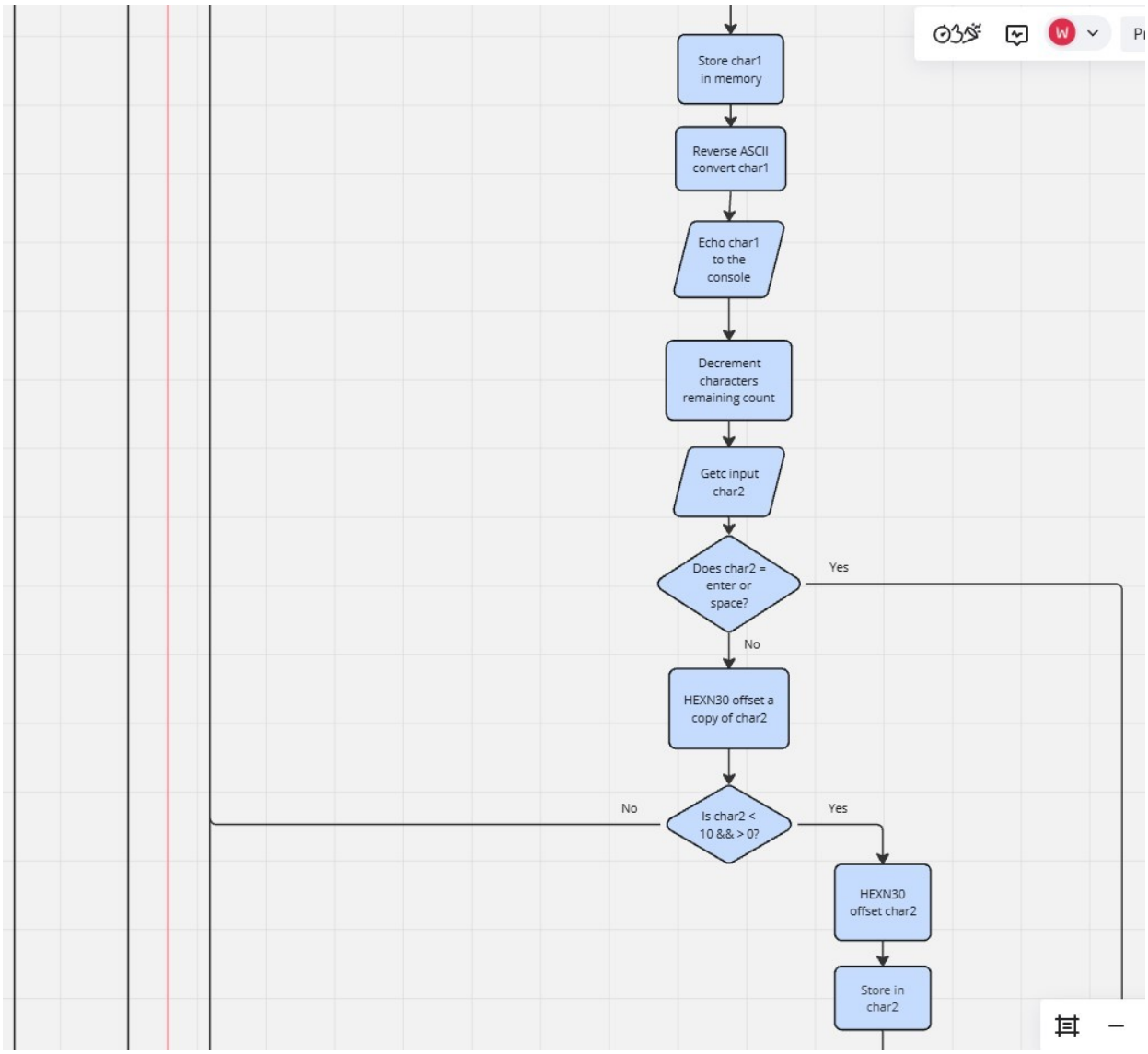
	stack is equivalent to a decimal number in the thousands.				
5/13/25	Figure out what happened in the most recent version and why it is only letting us enter one test score.	Ana Moreno	As stated above, the program needs to allow for entry of five test scores and progress from entering one tests score to the next in a way that is “natural” to the user.	High/ critical	In progress

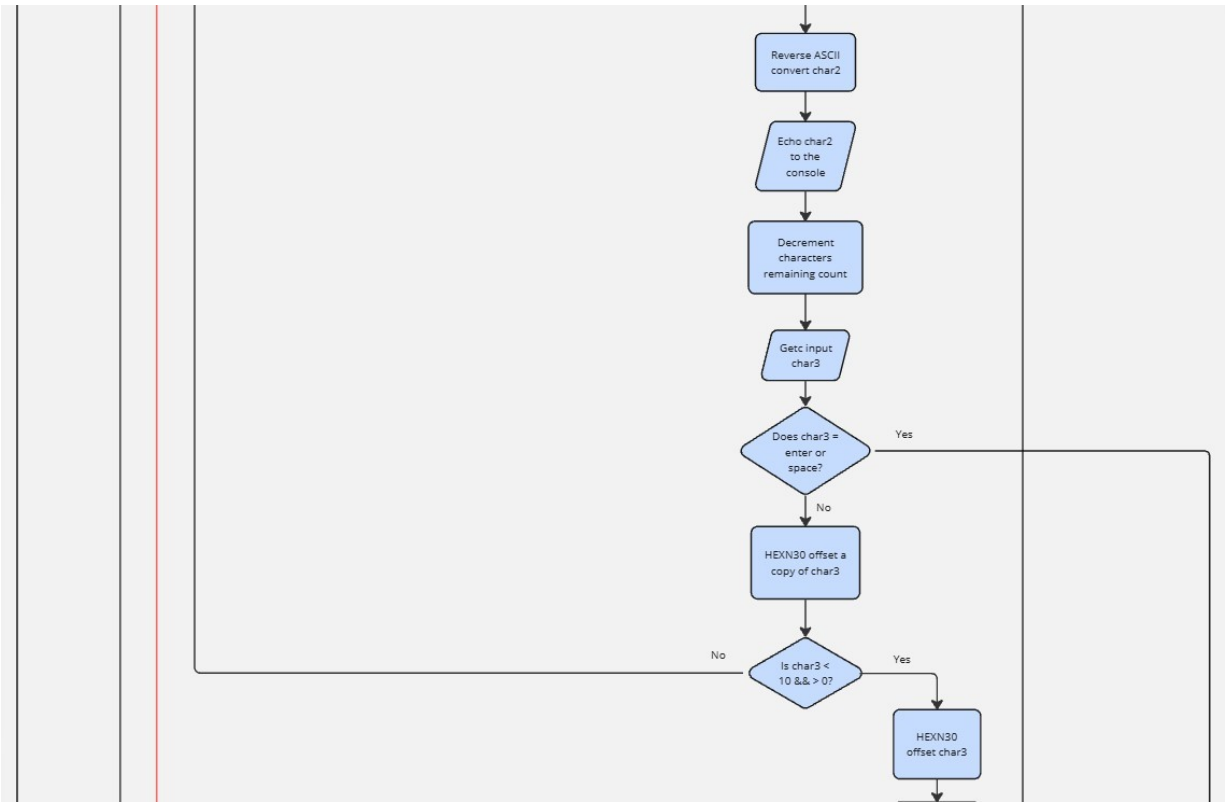
Part III – Appendices

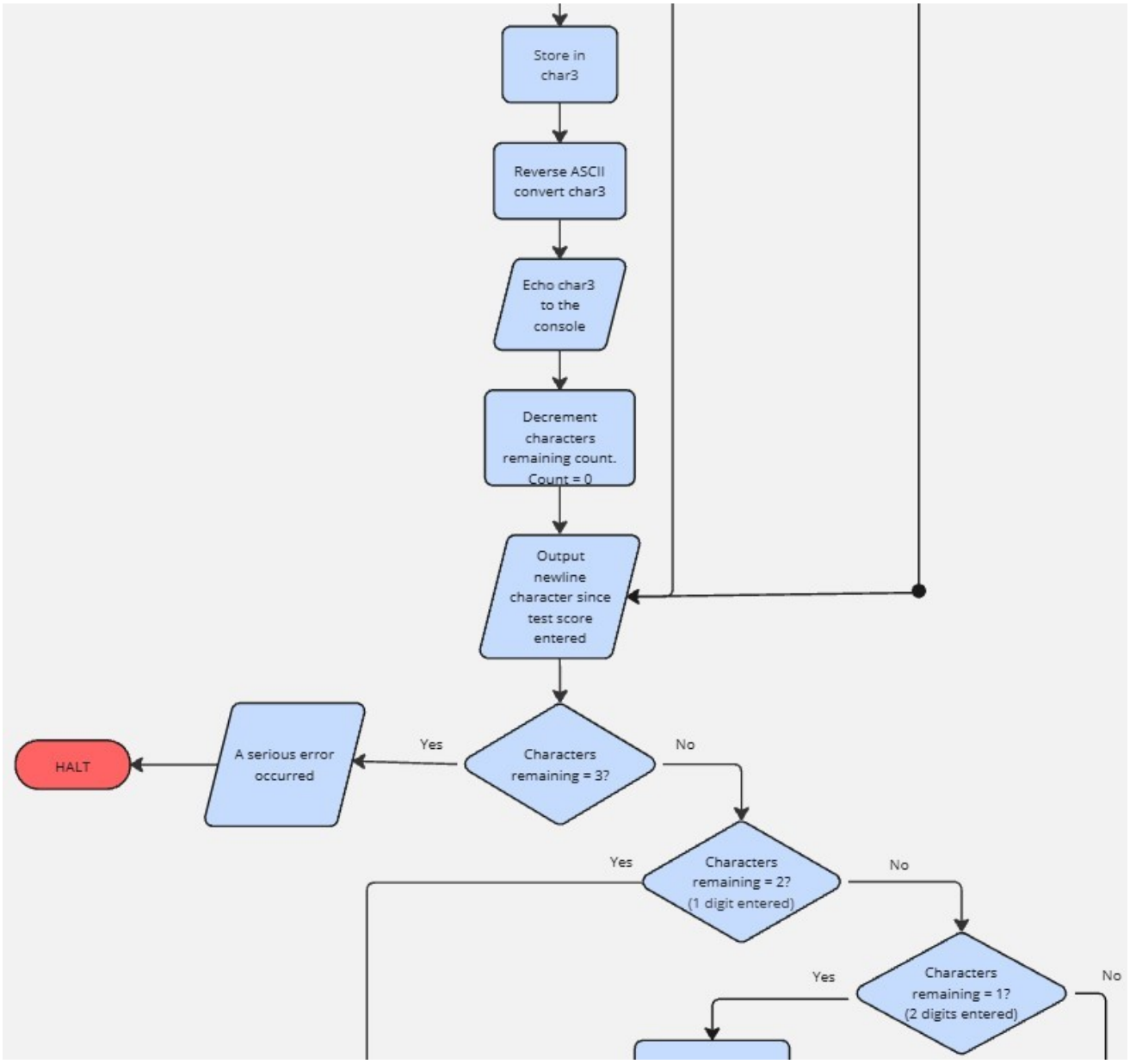
None

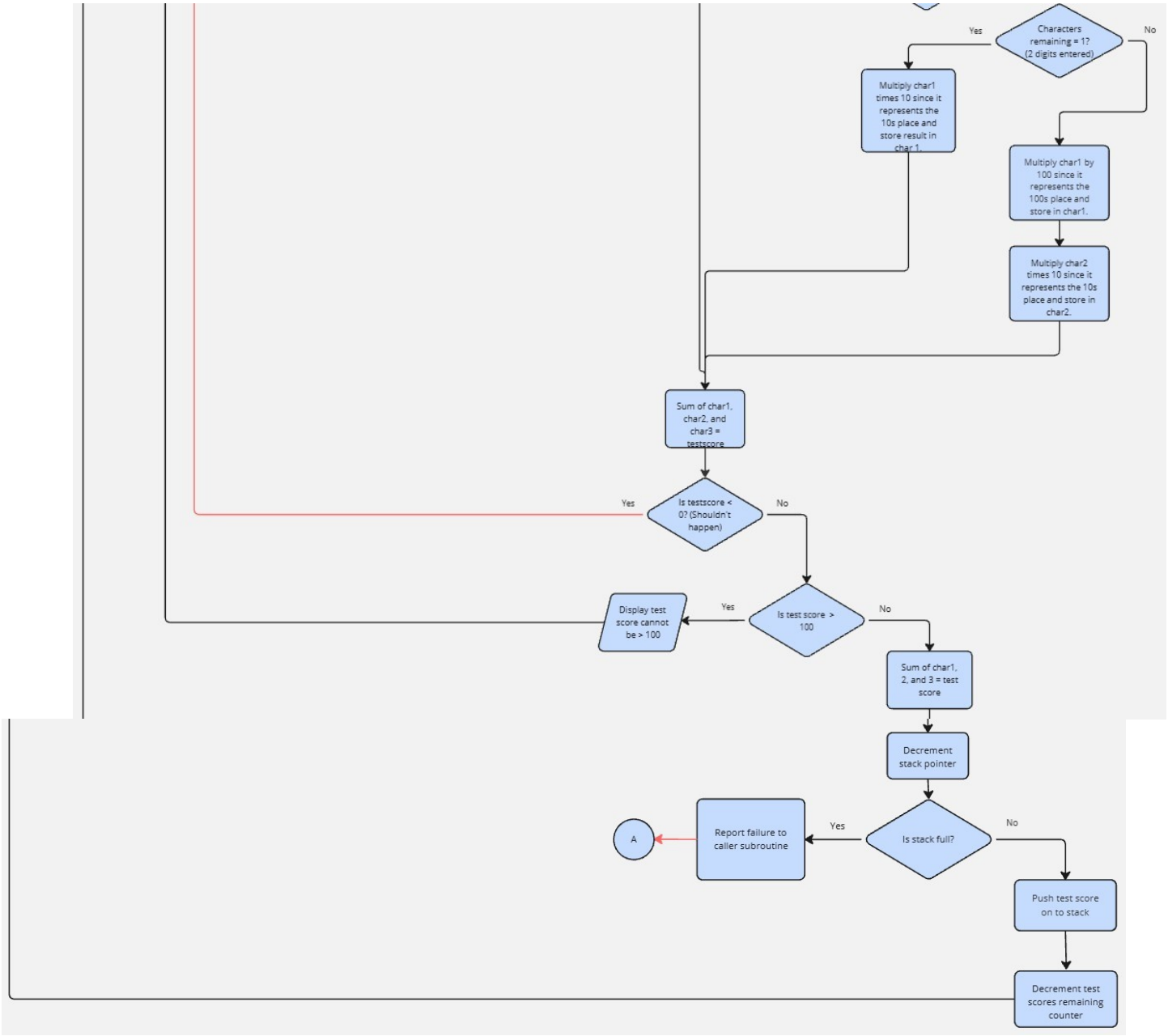
Flow chart or pseudo-code. Flowchart created using Miro.com

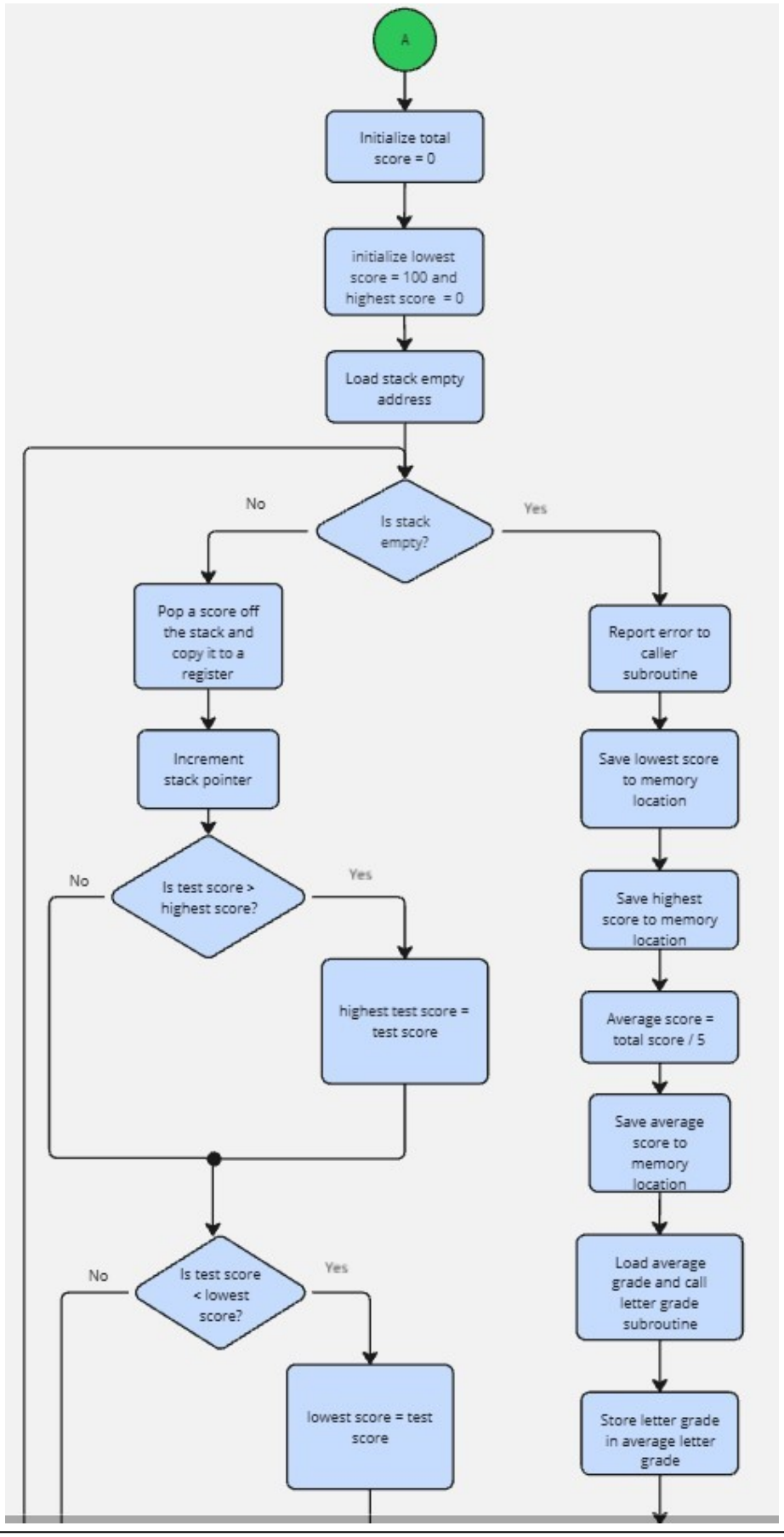


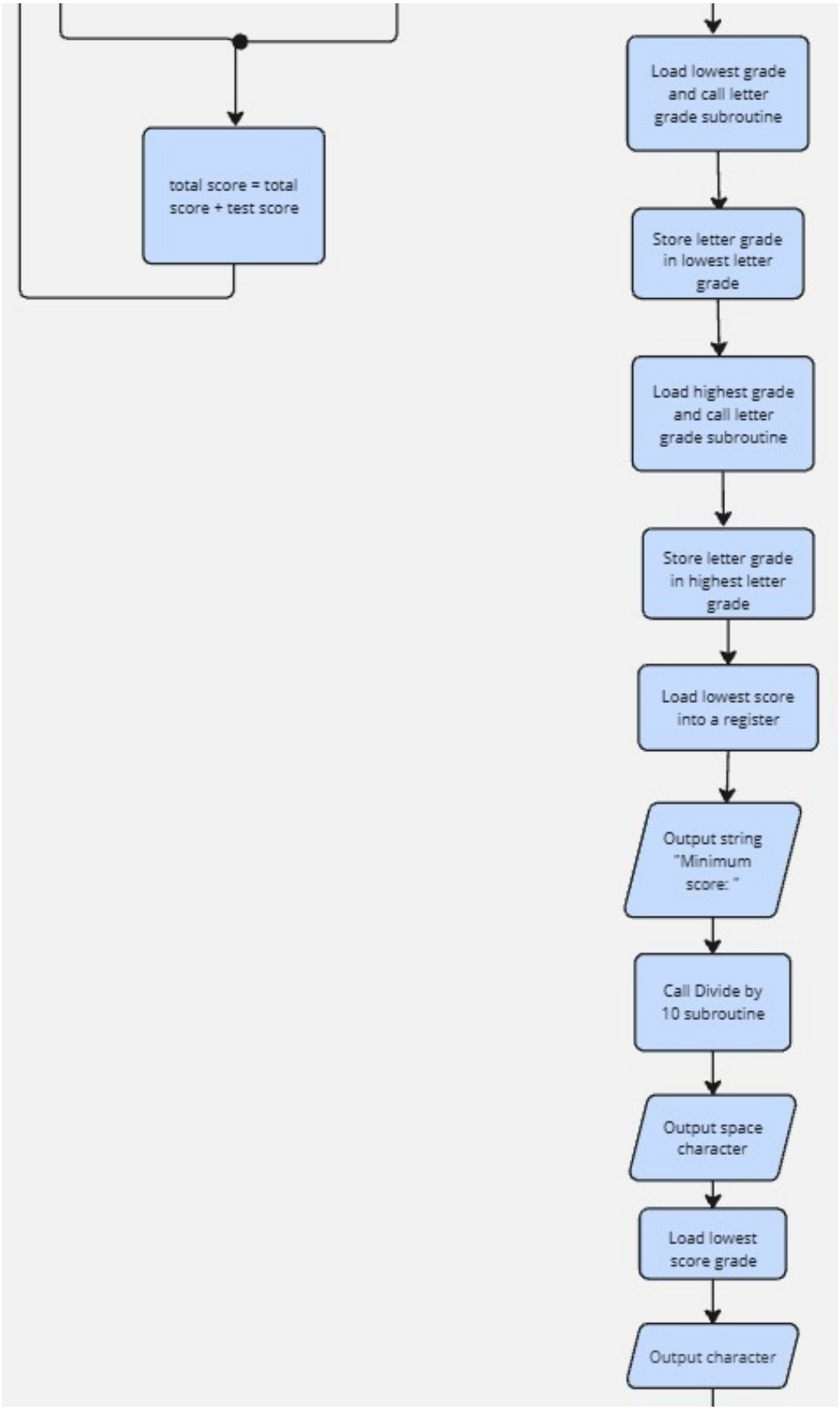

Store char1 in memory
Reverse ASCII convert char1
Echo char1 to the console
Decrement characters remaining count
Getc input char2
Does char2 = enter or space?
Yes
No
HEXN30 offset a copy of char2
Is char2 < 10 && > 0?
No
Yes
HEXN30 offset char2
Store in char2

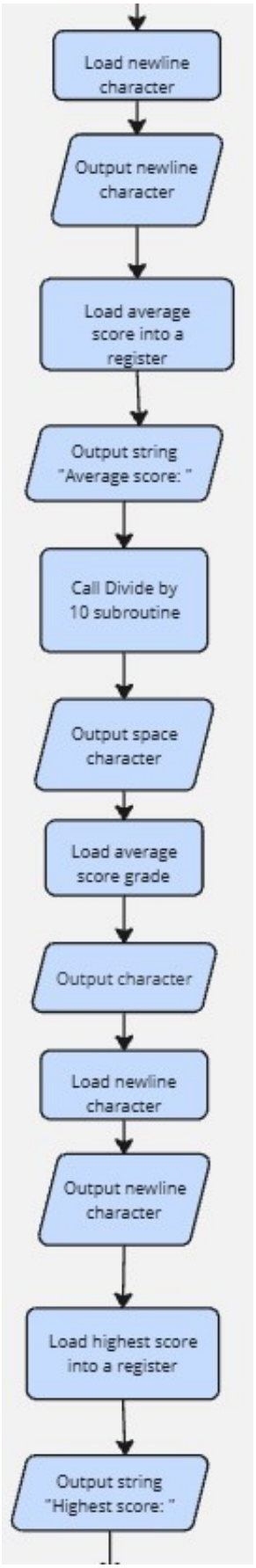

Reverse ASCII convert char2
Echo char2 to the console
Decrement characters remaining count
Getc input char3
Does char3 = enter or space?
Yes
No
HEXN30 offset a copy of char3
Is char3 < 10 && > 0?
No
Yes
HEXN30 offset char3

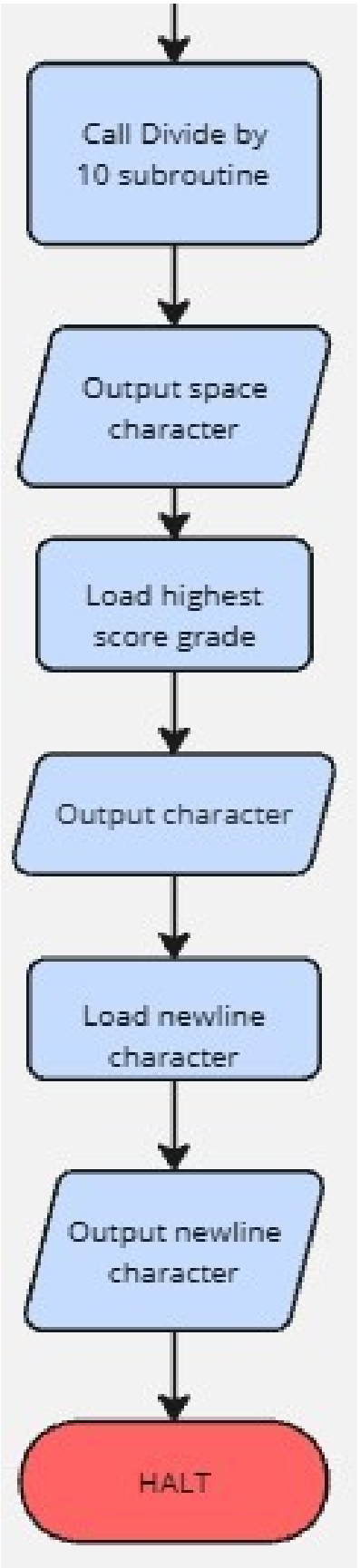

Store in char3
Reverse ASCII convert char3
Echo char3 to the console
Decrement characters remaining count. Count = 0
Output newline character since test score entered
Characters remaining = 3?
Yes
No
A serious error occurred
HALT
Characters remaining = 2? (1 digit entered)
Yes
No
Characters remaining = 1? (2 digits entered)
Yes
No

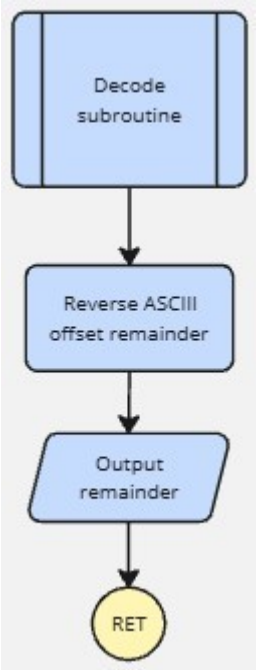

Yes
Characters remaining = 1? (2 digits entered)
No
Multiply char1 times 10 since it represents the 10s place and store result in char 1.
Multiply char1 by 100 since it represents the 100s place and store in char1.
Multiply char2 times 10 since it represents the 10s place and store in char2.
Sum of char1, char2, and char3 = testscore
Yes
Is testscore < 0? (Shouldn't happen)
No
Yes
Is test score > 100
No
Display test score cannot be > 100
Sum of char1, 2, and 3 = test score
Decrement stack pointer
A
Report failure to caller subroutine
Yes
Is stack full?
No
Push test score on to stack
Decrement test scores remaining counter

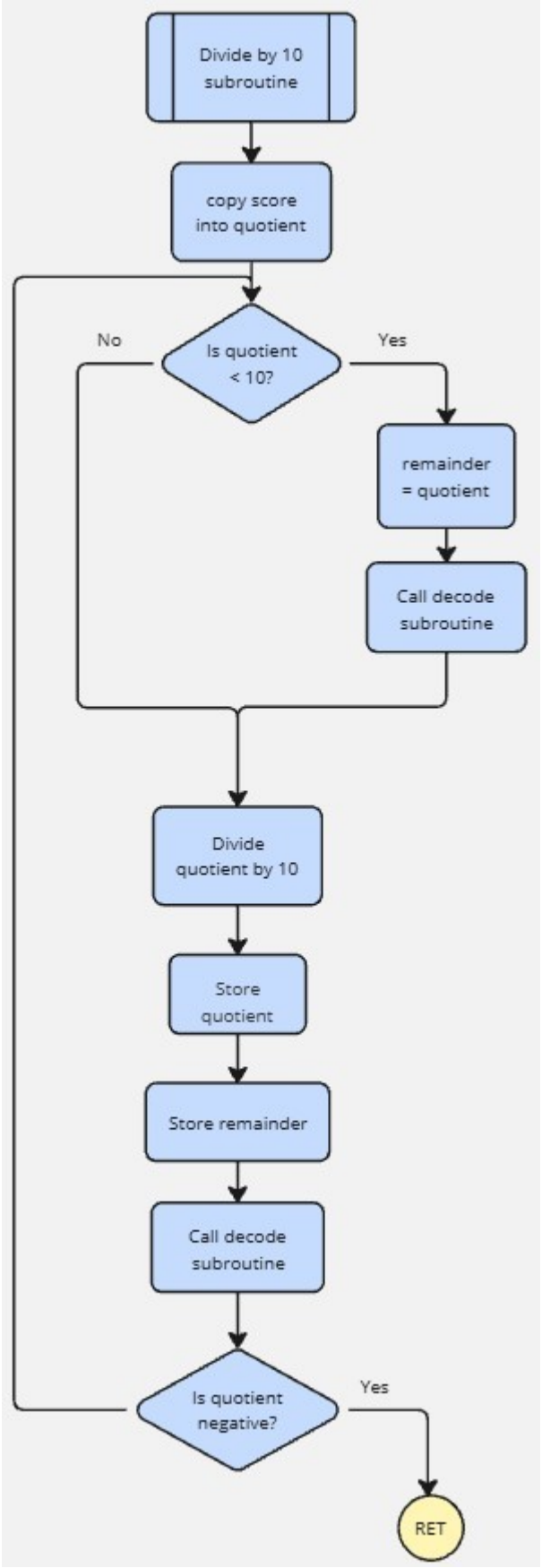

A
Initialize total score = 0
initialize lowest score = 100 and highest score = 0
Load stack empty address
Is stack empty?
No
Yes
Pop a score off the stack and copy it to a register
Increment stack pointer
Is test score > highest score?
No
Yes
highest test score = test score
Is test score < lowest score?
No
Yes
lowest score = test score
Report error to caller subroutine
Save lowest score to memory location
Save highest score to memory location
Average score = total score / 5
Save average score to memory location
Load average grade and call letter grade subroutine
Store letter grade in average letter grade

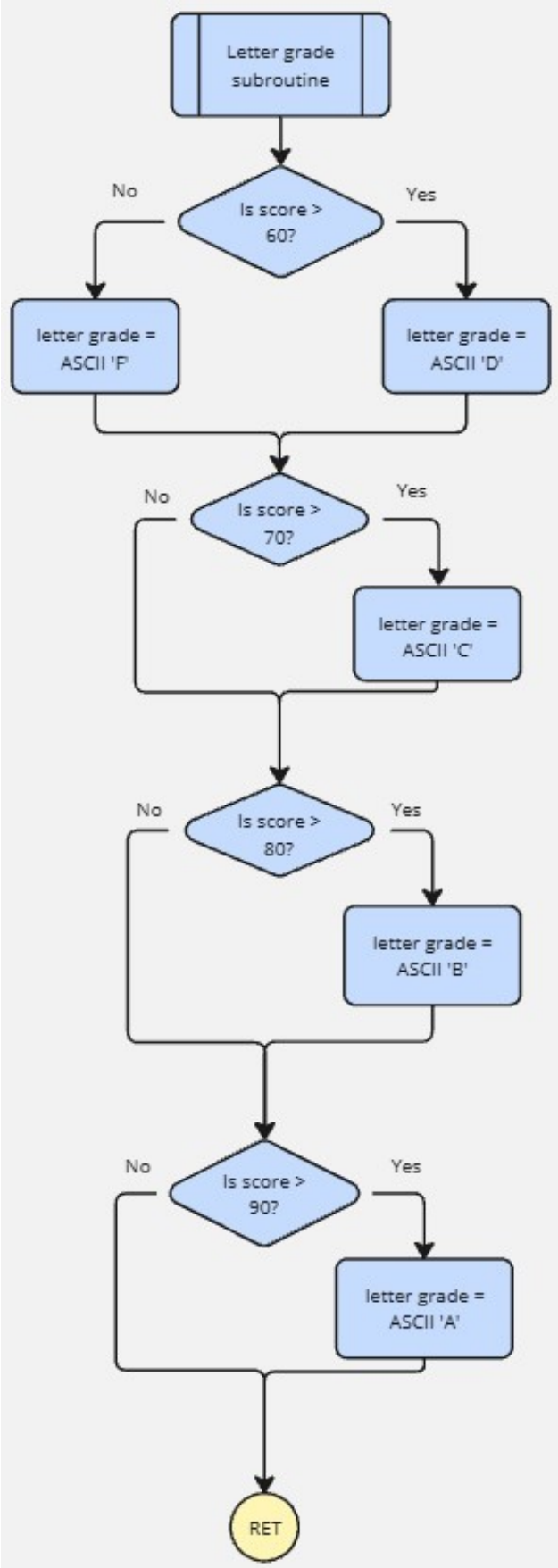

total score = total score + test score
Load lowest grade and call letter grade subroutine
Store letter grade in lowest letter grade
Load highest grade and call letter grade subroutine
Store letter grade in highest letter grade
Load lowest score into a register
Output string "Minimum score: "
Call Divide by 10 subroutine
Output space character
Load lowest score grade
Output character


Load newline character
Output newline character
Load average score into a register
Output string "Average score: "
Call Divide by 10 subroutine
Output space character
Load average score grade
Output character
Load newline character
Output newline character
Load highest score into a register
Output string "Highest score: "


Call Divide by 10 subroutine
Output space character
Load highest score grade
Output character
Load newline character
Output newline character
HALT


Decode subroutine
Reverse ASCIII offset remainder
Output remainder
RET


Divide by 10 subroutine
copy score into quotient
Is quotient < 10?
No
Yes
remainder = quotient
Call decode subroutine
Divide quotient by 10
Store quotient
Store remainder
Call decode subroutine
Is quotient negative?
Yes
RET


Letter grade subroutine
Is score > 60?
No
Yes
letter grade = ASCII 'F'
letter grade = ASCII 'D'
Is score > 70?
No
Yes
letter grade = ASCII 'C'
Is score > 80?
No
Yes
letter grade = ASCII 'B'
Is score > 90?
No
Yes
letter grade = ASCII 'A'
RET

; Course Project - Option B: Average 5 grades, display min, max, and average

.ORIG X3000

START

1. Set R2 to 0
2. Add 5 to R2
3. Store R2 in memory location SCORESENTERED
4. Load base address from BASE into R6
5. Load the prompt string address into R0
6. Print the prompt strings
7. Set R4 to 0
8. Store R4 in TESTSCORE
9. Store R4 in CHAR1
10. Store R4 in CHAR2
11. Store R4 in CHAR3
12. Add 3 to R4 (start with 3 characters to enter)
13. Get character input into R0
14. Set R1 to 0
15. Copy R0 to R1
16. Load -45 into R2
17. Add R2 to R1
18. If result is 0, go to CANTBENEG
19. Load -48 into R2
20. Add R2 to R1
21. If result is negative, go to BADCHAR
22. Load -58 into R2
23. Add R2 to R1
24. If result is 0 or positive, go to BADCHAR
25. Load -48 into R2
26. Add R2 to R1 (convert ASCII to digit)
27. Store R1 in CHAR1
28. Load 48 into R2
29. Load value from CHAR1 into R0
30. Add R2 to R0 (convert digit back to ASCII for echo)
31. Print character

32. Subtract 1 from R4 (now 2 characters left)
33. Get character input into R0
34. Set R1 to 0
35. Copy R0 to R1
36. Load -10 into R2
37. Add R2 to R1
38. If result is 0, go to VALIDNUMCHAR
39. Load -32 into R2
40. Add R2 to R1
41. If result is 0, go to VALIDNUMCHAR
42. Load -48 into R2
43. Add R2 to R1
44. If result is negative, go to BADCHAR
45. Load -58 into R2
46. Add R2 to R1
47. If result is 0 or positive, go to BADCHAR
48. Load -48 into R2
49. Add R2 to R1 (convert ASCII to digit)
50. Store R1 in CHAR2
51. Load 48 into R2
52. Load value from CHAR2 into R0
53. Add R2 to R0
54. Print character
55. Subtract 1 from R4 (1 character left)
56. Get character input into R0
57. Set R1 to 0
58. Copy R0 to R1
59. Load -10 into R2
60. Add R2 to R1
61. If result is 0, go to VALIDNUMCHAR
62. Load -32 into R2
63. Add R2 to R1
64. oif result is 0, go to VALIDNUMCHAR
65. Load -48 into R2
66. Add R2 to R1

---

67. If result is negative, go to BADCHAR
68. Load -58 into R2
69. Add R2 to R1
70. If result is 0 or positive, go to BADCHAR
71. Load -48 into R2
72. Add R2 to R1 (convert ASCII to digit)
73. Store R1 in CHAR3
74. Load 48 into R2
75. Load value from CHAR3 into R0
76. Add R2 to R0
77. Print character
78. Subtract 1 from R4 (now 0 characters left)
79. Load newline character into R0
80. Print newline
81. jump to subroutine PREPNUMCHAR
82. Subtract 3 from R3
83. If result is 0 or positive, go to CONTINUE

84. Else, go to CHARQTYERR
85. Jump to subroutine CALCULATESCORE
86. Jump to subroutine DISPLAYRESULT
87. HALT
88. Load address of BADCHAR_MSG into R0
89. Print BADCHAR_MSG
90. HALT
91. Load address of NEGATIVE_MSG into R0
92. Print NEGATIVE_MSG
93. HALT
94. Load address of CHARQTY_MSG into R0
95. Print CHARQTY_MSG
96. HALT
97. Label PREPNUMCHAR
98. Load CHAR1 into R1
99. Multiply R1 by 100 (shift left, then add)
100. Load CHAR2 into R2
101. Multiply R2 by 10 (shift and add)

102. Add R2 to R1
103. Load CHAR3 into R3
104. Add R3 to R1
105. Store R1 into TESTSCORE
106. Return from subroutine
107. Label CALCULATESCORE (RET)
108. Load TESTSCORE into R1
109. Load SCORESENTERED into R2
110. Add R1 to R2
111. Store result in SCORESENTERED
112. Return from subroutine (RET)
113. Label DISPLAYRESULT
114. Load address of RESULT_MSG into R0
115. Print RESULT_MSG
116. Load SCORESENTERED into R1
117. Convert R1 to ASCII string (loop or call helper if needed)
118. Print value in R1 (character-by-character)
119. Load newline into R0
120. Print newline
121. Return from subroutine (RET
122. Fill labels W/ placement numbers

Works Cited

Patt, Yale, and Patel, Sanjay. *Ise Introduction to Computing Systems: From Bits & Gates to C*. 3rd ed., McGraw-Hill, 2020.

"What Is a Swimlane Diagram?" *Lucidchart*, 8 May 2025, www.lucidchart.com/pages/tutorial/swimlane-diagram#:~:text=A%20swimlane%20diagram%20is%20a,employee%2C%20work%20group%20or%20department