CIS-11 Project Documentation Template

Team DJ
Daniel Rios, Jade Thong
Test Score Calculator
May 19, 2025

Advisor: Kasey Nguyen, PhD

Part I – Application Overview

Objectives

The Test Score Calculator project is designed to improve the efficiency and accuracy of processing test score data. This program allows users to input five test scores, then automatically calculates and displays the **maximum**, **minimum**, and **average** scores, as well as the corresponding **letter grade** based on a predefined grading scale.

Why are we doing this?

To elicit the objectives, ask the business expert, the development manager, and the project sponsor the following questions:

- What business objectives of the company will this project help achieve? Possible objectives might be reducing costs, improving customer service, simplifying the workflow, replacing obsolete technology, piloting a new technology, and many others. Also, make sure you understand exactly how the proposed project will help accomplish the stated objective.

  The business objectives of the company that this project will help achieve are improving efficiency, simplifying workflow, and enhancing reliability and accuracy. The project improves efficiency by eliminating the need for manual input and computation of grades. This reduces human error and speeds up the grading process, leading to greater efficiency. By using a standardized program to process the test scores, the project ensures consistent and error-free results. Time and effort used to calculate test scores can be redirected elsewhere.

- Why are we doing this project now? What will happen if we do it later? What if we do not do it at all?

  We are creating this project now to streamline the process of grading tests. As we transition to a digitized age, there is a need for simplicity and efficiency. In academia, administrators require custom solutions that align with their specific grading standards, and this program will reduce workload by automating score interpretation while minimizing error. Administrators and teachers can use the time they would spend on grading tests on other tasks of greater importance. If we continue to delay this project, we will continue to rely on manual input, which increases the probability of human error and inefficiency. In terms of cost effectiveness, more time is spent on grading tests and menial work, which can be cost inefficient. If this project is not done at all, the company would continue to invest time and resources in manual solutions that would not align with cost goals, indicating a loss in profit and work done.

- Who will benefit from this project? Do the people who will benefit from it consider it the most important improvement that can possibly be made at this time? Should we be doing a different project instead?

  Administrators, teachers, and students will all benefit from this project. Academic institutions seek reliable tools to manage the manual burden of grading while improving accuracy and consistency. The people who benefit from this project the most would consider it the most improvement that could possibly be made for this time, as this project targets a fundamental task in education– test grading. This has a high impact on workflow efficiency and user satisfaction, and it is an initiative that can be implemented immediately, encouraging efficiency and speed. Therefore, this project is the right one to prioritize. Compared to a bubble sort and a character counter, this project directly relieves the burden of test-grading.

Business Process

Currently, test score evaluation is performed manually by educators or administrators. This process involves collecting the test scores, calculating the average, and determining the appropriate letter grade based on a grading scale. These steps are usually carried out on pen and paper or by using calculators and spreadsheets such as Excel.

This manual approach can be time-consuming and has a high probability of human error. Grading consistency can vary depending on who performs the calculations and educators must double-check their work frequently to ensure accuracy or, in cases where the mistake is overlooked, will return a test score that is incorrect. As the number of tests increases, this process becomes more inefficient and places a significant burden on time and resources.

With the implementation of the Test Score Calculator, the grading process will become automated and standardized. Educators will input five test scores directly into the program, and the program will automatically calculate the minimum, maximum, and average scores while also determining the appropriate letter grade. This tool eliminates the need for manual computation, significantly reduces the possibility of human error, and ensures that each test is applied to the grading scale. This process will become faster and more accurate and less labor-intensive, freeing up time for educators to focus on higher-priority tasks such as student engagement and curriculum management.

User Roles and Responsibilities

The Test Score Calculator is a tool designed to assist the evaluation of test scores by automating calculations and assigning grades for each test. The system is simple but effective, serving multiple user roles.

The first user role is the instructor or educator. The primary business objective is to ensure timely, consistent, and accurate evaluation of test scores. Their tasks in the system are to input the test scores, interpret the output, record the results, and use the output data to identify trends and outliers. This will be performed regularly after each test or quiz, which depends on the frequency of evaluations teachers give. They will then hand-off to the administrator.

The second user role is the administrator or academic coordinator. They ensure consistent grading practices throughout each class and monitor academic performance to make sure each student is on track. Their tasks in the system are to review output from instructors, provide support for instructors, and analyze score distributions for reporting and policy decisions. These tasks will be performed on a monthly basis, which they will hand off to district stakeholders or data systems.

Production Rollout Considerations

The Test Score calculator will follow a deployment schedule in waves to ensure a

smooth transition and to minimize disruptions for existing processes and allow time for feedback. Firstly, the programmers will ensure the system is functional and ready for use before developing a brief user guide and a reference sheet for instructors and administrators, outlining how to input and interpret test results. The program will be deployed to a small group of instructors first to provide early feedback and report any issues. Upon successful initial testing, the program will be released to all instructors with a contact point for ongoing technical support for users who run into errors. After the deployment, there will be a collection of data for the programmers to analyze the system's performance and utility. If there are errors, the programmers will release an updated version of the program and update training materials based on the improved Test Score Calculator as needed. Since the program does not rely on storage, no data migration is needed. The program operates entirely on user input during each session. The expected input per session is five test scores and the expected use frequency is once per test. The expected number of users is a small group (less than 15) in the initial phase and will expand to the entire department after the rollout. Each transaction is instantaneous because the program runs locally and uses minimal system resources.

Terminology

Test Score Calculator – A software application that accepts five numerical test scores as input and automatically calculates the maximum, minimum and average scores.

Test Score – A numerical value that represents how a student performed on a test/quiz.

Letter Grade – A representation of a student's performance in letter form, i.e. A, B, C, D, F.

Grading Scale – A system used to assign grades to student work, in either numerical or letter form.

Minimum Score (max) – The lowest value of the five test scores input into the program.

Maximum Score (min) – The highest value of the five test scores input into the program.

Average Score (avg) – The mean of the five test scores, calculated by summing the scores and dividing by the number of scores.

I**nstructor** – The primary user of the program, responsible for entering test scores and interpreting the results for grading.

Administrator – A user who supports and oversees the grading process across multiple classes and departments, ensuring consistency for reporting.

Standardization – The process of making something conform to a standard.

Rollout – The deployment of the Test Score Calculator, beginning with a

limited initial testing phase with feedback and expansion.

Transaction – In which a user runs the program (inputting five test scores and receiving an output).

Session – A single use of the program, during which one set of test scores is processed.

Data Migration – The transfer of data from one system to another.

Deployment – The release and installation of the software to the public.

Efficiency – The ability to produce a desired output with a minimum of effort or expense.

Consistency – Conformity in the application of something.

Reliability – To perform consistently well; trustworthy.

Human Error – Any mistake made by a human, caused by a variety of factors, e.g., fatigue, lack of training, flawed design.

Accuracy – Correct or precise.

Streamline – To make a system more efficient and effective by employing faster and simpler working methods.

Part II – Functional Requirements

Statement of Functionality

The Test Score Calculator application will allow a dedicated user like an instructor to input five numerical test scores from a range of 0 to 100 percent of the total score. When the test scores are received the program will function accordingly:

1. Storing input: Program will prompt the user for five individual scores and store them into a designated memory array.

2. Processing Data: The program will identify and store the maximum, minimum, and average of the input scores. The score analysis will be managed through arithmetic logic like addition, subtraction and the use of comparison operations like "NOT" for example.

3. Letter Grade Equivalence: In handling the data, the program will also account for the letter grade equivalent of the input scores. Meaning, it will also show the corresponding letter grade based on the earlier scale where:

- 90-100% = A
- 80-89% = B
- 70-79 = C
- 60-69 = D
- 0-59 = F

4. Displaying data: The program will output the maximum, minimum, and average of the input scores alongside the letter grade equivalent of the scores.

5. Control Flow: The Test Score Calculator will demonstrate control flow through the use of iteration/looping to handle and evaluate the input scores. Conditional branching will also be utilized to aid in determining/comparing the maximum, minimum and also the assignment of the letter grades to the number scores. Additionally, appropriate system call directives will be used to manage both input and output operations.

6. Modular Subroutines: The program will be able to handle critical functions like finding the max/min, calculating the average, ASCII conversions, general arithmetic operations, and processing the input through dedicated subroutines. Any additional subroutines may be implemented as needed for assisting in handling these main tasks effectively.

7. Stack Management: Will include PUSH-POP operations to retrieve and handle data throughout the subroutines while also accounting for save-restore operations to effectively preserve the programs' state in the subroutines.

8. Managing Memory: The application will utilize specific memory addresses to allocate arrays, constants, and stack pointers to avoid overflow.

9. Test Run: The program will be tested with the specified values of 52, 87, 96, 79, and 61 to ensure the output correctly displays the minimum, maximum, average and letter grade.

10. User Interface: The program will prompt the user as mentioned for 5 test scores and will provide a score analysis showcasing the above listed features in a neatly formatted manner. Having mainly an instructor in mind as the dedicated user category, it will allow for a quick and feasible way to calculate/manage grading in a casual capacity (only being able to handle a handful of scores/data).

Scope

The Test Score Calculator project will be developed and delivered in a single phase, designed to process five test scores per session. The project focuses on computational accuracy and usability and the public will receive immediate functionality when the project is published. We plan to have an initial testing group of a contained number of educators who have access to this tool and after receiving feedback and adding bug fixes, the program will be released to the public. Future expansions could include data export, customization of grading scales, and the program accepting more than five scores, but the initial scope prioritizes a reliable tool for immediate use.

Performance

Performance requirements are as follows:
Calculations and output display must be completed within 1 second of input submission.
Input prompts must appear within 0.5 seconds of the program starting.
The program must process 5 scores per transaction with 100% accuracy per session.
Stack operations (PUSH/POP) must avoid overflow.

Usability

The usability requirements for the Test Score Calculator consist of input simplicity, output clarity, and error handling. The user must be able to enter all five test scores with a single action to process (pressing “Enter”) and the program must reject non-numeric input or values outside the valid range of 0-100. The results of the program should be clearly labeled and presented in a readable format for clarity and ease of readability.

Documenting Requests for Enhancements

Date	Enhancement	Requested by	Notes	Priority	Release No/ Status
5/19/2025	Allow for greater input customization than just being limited to 5 test scores	Instructor user Category	Being able to calculate averages for scores above 5 would allow for more large-scale and practical uses inside real-world classes that are typically more than 5 students.	High	Upcoming Version 1.1 update
5/20/2025	Allow user to export data results to a file	Admin Coordinator	Saving data results to be able to document test scores	Low	Backlog
5/25/2025	GUI improvement instead of just using a console input/output	IT Team	Improvement of the UI is generally more user friendly, and the neatness of the application will make it more appealing	Low	Version 2 Planning

Part III – Appendices

Flow chart or pseudo-code.

Include branching, iteration, subroutines/functions in flow chart or pseudocode.

< < < START OF PROGRAM > > >

1. Output user prompt, asking for the input of 5 different test scores (0-100).

2. Account for newline characters / output formatting for program neatness

3. Repeat 5 times:
    a. Call function to retrieve store (unofficial name: (GET_SCORE):
        - Read two characters from user input
        - Convert ASCII digits to an integer (supporting 0-100)
        - Store result in a temporary register (e.g., R3)
    b. Store score inside an array that contains all of the grades: “GRADES”
    c. Call function to compare the score to the letter grade scale
        - Determine & store the corresponding letter grade
    d. Output both the score & letter grade
        - Print int score
        - Output space and the letter grade equivalent of each score
    e. Call Stack Management function to:
        - PUSH score to stack using pointer, essentially decrementing stack pointer and storing it for future use
    f. Print newline / create space

4. Call function to find the maximum of scores (unofficial name example: FIND_MAX):
    a. Loop through GRADES array
    b. Track the highest value in MAXVAL & store
    c. Print/output "Maximum: " followed by the calculated max score

5. Call function to find the minimum of the scores (example: FIND_MIN):
    a. Loop through GRADES array
    b. Track the lowest value in MINVAL & store
    c. Print/output "Minimum: " followed by the calculated min score

6. Call function to find the average of the minimum scores (example: FIND_AVG)
    a. Loop through GRADES array and find the total sum of the scores using repeated a chain of arithmetic logic like addition.
    b. Divide total by 5 to find the average, using repeated subtraction
    c. Print "Average: " followed by the calculated average of the scores
    d. (optional): Compare average score to the grading scale of:
        - 90–100 -> A
        - 80–89 -> B
        - 70–79 -> C
        - 60–69 -> D
        - 0–59 -> F
        - Print "Letter Grade: " and the corresponding letter accordingly

7. (optional display POP usage & pop one score off the stack & print before halting)

8. HALT the program

< < < END PROGRAM > > >