Milestone 3 Scrum Report

All students are expected to attend the scrum meetings and to participate. Failure to do so will result in greatly reduced grades.

GROUP: 10

Members Present:

1. Duong Truong Phuc Nguyen	4. Ahnaf Tahmid Khan
2. Jasmin Aro	5.Syed Abdullah
3.Huynh Huy Hoang	6.

Milestone 3 Tasks

In this milestone you will create issues to design the functions, design all of the functions you need to complete the project and store the specifications in the repository. As soon as the specifications start to be produced, you can start to design the blackbox tests (what they test, how to perform them and test data). Once tests are written, they can be implemented and added to the repository and any team members not otherwise busy can start to implement the functions. You will also build a function-test matrix that shows the blackbox tests for each function. This will be maintained through the testing cycle as new tests are added.

Deliverables due 4 days after your lab day:

- A set of AT LEAST 4 function specifications added to a new header file and stored in the repository.
- A set of blackbox tests as test documents (in an Excel file) with test data for the functions you created. At least 4 sets of test data are required for each function. You must have test cases for at least 6 functions (including all your custom functions). Stored in the repository.
- **Create and add a C++ testing project to your solution.**
- Start writing blackbox test code (for the functions above) and store in repository (at least 1 is required for this milestone).
- Start implementing the functions and store them in repository (optional).
- A requirements traceability matrix added to the repository and shows the mapping between the requirements and test cases.
- Updated Jira project to show activities and progress.
- Completed scrum report including reflection questions answered.

Rubric:

Individual	Group participation (includes GitHub commits and Jira usage)	80%
	Teamwork	20%
Group	Function specifications (documented, complete, well-written, added to the project)	10%
	Blackbox test cases document (well-written, complete, good test data)	10%
	Blackbox test code (in the C++ project) well-designed and documented	10%
	Functions implementation (coded in the C project & well documented)	10%
	Visual Studio solution with 2 projects (complies and works)	10%
	Requirements traceability matrix (complete and added to GitHub)	10%
	Git usage (used properly with good structure)	10%
	Jira usage (creates issues, tracks progress)	15%
	Scrum report & reflections	15%
Deadline	20% deduction for each day you are late	

Scrum Report

Summary of Tasks Completed or Delayed in the last week:

Here you can list all of the tasks completed in the last week along with any tasks which could not be completed with a reason why they could not be completed.

Member	Tasks Completed	Tasks Delayed/Blocked
Ahnaf Tahmid Khan	**reflection 1 , reflection 2,**	**N/A**
Jasmin Aro		**unit testing eqPt() addPackage()**
Duong Truong Phuc Nguyen	**reflection question 2 + create header file with data structures + update the test plan in 5. , 10. , 11., 12., 14., 15., 16., 17.**	**N/A**
Huynh Huy Hoang	**manage the data sent in github repository and submit the work**	**N/A**
Syed Abdullah	**Did tests for functions double distance and double calculateCapacity, Created Black box test code for calculate**	**N/A**

	capacity function, created the traceability matrix, updated Jira	

For every task delayed or blocked, describe the reason for the delay or block, how it impacts the project and the proposed solution or workaround**.**

Delayed or Blocked Task	**unit testing eqPt() addPackage()**
Reason for delay or block	**catching up with other assignments**
Impact on Project	
Solution or work-around	**will continue unit testing on the weekend before Monday scrum**
Delayed or Blocked Task	
Reason for delay or block	
Impact on Project	
Solution or work-around	

Summary of Meeting:

A summary of the main points discusses in the meeting and the outcomes of the discussions.

Topic	**Discussion Summary**	**Outcome**
Analysis	**Understanding the requirements of this week and deciding distribution of work**	
Work Division	**Discussed who would work on what**	**Work was divided up evenly**

Summary of Decisions Made:

This will include major architecture and design decisions, testing decisions, prioritization of tasks, dealing with problems encountered and other major outcomes from the meeting.

Decision	Rationale
Division of Work	Everybody should have a equal amount of work

Tasks Attempted During Meeting:

Each member is assumed to participate in the scrum meeting and contribute to the completion of the scrum report and reflections. Since the scrum meeting will not take more than 20-30 minutes, there is lots of time left to undertake some of the actual work tasks. In the table below, each member should list what they did to complete the scrum report, the reflections, and 1-4 other tasks they completed during the class period. If a task could not be completed, the student should indicate why this was not possible.

Member	Task Attempted	Time Spent	Complete ?
Jasmin Aro	**unit testing eqPt() and addPackage()**		**no**
Ahnaf Tahmid Khan	**reflection 1 , reflection 2**	**40 mins**	**yes**
Duong Truong Phuc Nguyen	**create set of functions and add to new header file and add to GitHub, 2 test data with 20 test cases, implementing the function and store in repository, created and add C++ testing project**		**yes**
Syed Abdulah	**Fill in my part of scrum report, Did tests for functions double distance and double calculateCapacity, Created Black box test code for calculate**	**20**	**yes**
Huynh Huy Hoang	**reflection 3 + testing functions getNumRows, getClosestPoint and eqPt**	**2 hours**	**yes**

Scrum Tasks Selected for Next Week:

The tasks each member has selected to pursue for this class or the next week.

Group Member	Task Description
Everyone	Scrum meeting to discuss division of work

Major Outcomes of Meeting:

This is where you should highlight the major accomplishments of the class.

Outcome	Impact on Project
Division of work	**Everyone does their part so that a complete solution is created**

Things That Went Well in This Meeting:

Here you can highlight things which worked well. This indicates that the way you worked on these items is working and should be continued.

Topic/Work Item	Reason for Success
Attendance	**Everyone showed up**

Things That Did NOT go Well in This Meeting:

This is where you can list things which did not go well in the class. You should analyze why this happened and suggest how you can improve it next time. This will lead to the goal of *continuous process improvement*.

Topic/Work Item	Reason for Problem and How to do Better

Reflections:

Answer the following questions using your own words. Make sure that each answer comprises a minimum of 100 words.

1. What is the difference between blackbox tests cases and blackbox test code? Explain how we use assertion in Visual Studio to execute tests.

   Blackbox test cases specify inputs and anticipated results in software testing to assess program performance from an external viewpoint, without knowledge of the internal code structure. In contrast, the actual code developed to execute these test cases within a testing framework is referred to as blackbox test code. Typically, assertions are employed to verify outcomes. To compare expected results with actual outputs, Visual Studio utilizes assertions like Assert.AreEqual. For instance, the assertion Assert.AreEqual(expected, result) will pass when testing an addition function if the output matches the expected value, indicating that the code operates correctly; if it does not, the test is marked as failed, highlighting any discrepancies.

2. How can a traceability matrix help in the testing process?

A crucial tool in the testing process is the traceability matrix, which aids in identifying any gaps in test coverage by ensuring that all requirements are addressed through corresponding test cases. This matrix minimizes the risk of overlooking essential functionality during testing by clearly indicating which requirements have been tested and which have not, thereby linking requirements to their respective test cases. Furthermore, it allows testers to monitor changes in requirements and assess their impact on test cases, facilitating timely updates and alignment throughout the testing process. Additionally, the traceability matrix is beneficial for audits and overall quality assurance, as it serves as evidence that each criterion has been validated, promoting accountability.

3. Write down two of the function prototypes you submitted. Why did do you need each one of them and how will each one help you achieve the project needs?

The function that I submitted:

```
int canAddPackage(const struct Truck* tr, const struct Package* pkg);
int countPackages(const struct Truck* tr);
```

These functions help our group testing data easier and more effectively. The function “int canAddPackage(const struct *tr, const struct Package * pkg);” checks if there is enough weight and volume left before adding more data into them. It’s return 0 if the data input (weight and volume) is less than 0, and return 1 if there is enough to add the data. The “int countPackages(const struct Truck* tr);” functions returns the package of the truck, this also help us to track the number of packages for other testing functions .