./

Learning Report –

SDLC



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **Approved By** | **Remarks/Revision Details** |
| 1.0 |  | Sneha Anand |  | PSP |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Document History**

# 

Contents

[ LIST OF FIGURES 5](#_Toc68882040)

[1. SDLC: Software Development Life Cycle 6](#_Toc68882041)

[1.1 SDLC models: 7](#_Toc68882042)

[2. ACTIVITES 14](#_Toc68882043)

[Activity 1: 14](#_Toc68882044)

[ Task 14](#_Toc68882045)

[ Requirement 14](#_Toc68882046)

[ Explanation 14](#_Toc68882047)

[Activity 2a: 16](#_Toc68882048)

[ Task 16](#_Toc68882049)

[ Explanation: 16](#_Toc68882050)

[Activity 2b: 17](#_Toc68882051)

[ Task 17](#_Toc68882052)

[ Explanation 17](#_Toc68882053)

[Activity 2c: 19](#_Toc68882054)

[ Task 19](#_Toc68882055)

[REQUIREMENTS 19](#_Toc68882056)

[BRIEF HISTORY OF A CALCULATOR 19](#_Toc68882057)

[20](#_Toc68882058)

[PRODUCT DEFINITION 21](#_Toc68882059)

[1(A). OLD FEATURES OF THE DEVICE 21](#_Toc68882060)

[1(B). FUNCTIONS THAT CAN PERFORMED FROM THE DEVICE 21](#_Toc68882061)

[1(C). FUNCTIONS THAT WE WILL INCLUDE IN OUR DEVICE 22](#_Toc68882062)

[1(D). TARGET AUDIENCE 22](#_Toc68882063)

[4W-1H 24](#_Toc68882064)

[What 24](#_Toc68882065)

[When 24](#_Toc68882066)

[Where 24](#_Toc68882067)

[Why 24](#_Toc68882068)

[How 24](#_Toc68882069)

[Valid inputs for our device are 24](#_Toc68882070)

[HIGH LEVEL AND LOW LEVEL REQUIREMENTS 25](#_Toc68882071)

[Procedure to be followed if multiple functions are selected together 28](#_Toc68882072)

[High Level design 29](#_Toc68882073)

[Structural diagram 29](#_Toc68882074)

[Behavioural Diagram 30](#_Toc68882075)

[LOW LEVEL DESIGN 31](#_Toc68882076)

[Structural Factorial 31](#_Toc68882077)

[Behavioural Factorial 32](#_Toc68882078)

[High Level Test Plan 33](#_Toc68882079)

[LOW LEVEL TEST PLAN 34](#_Toc68882080)

[SCOPE OF TESTING 34](#_Toc68882081)

[Implementation 35](#_Toc68882082)

[Github Repositories 39](#_Toc68882083)

# LIST OF FIGURES

[Figure 1:SDLC 5](#_Toc66919448)

[Figure 2:waterfall model 6](#_Toc66919449)

[Figure 3:Agile Model 8](#_Toc66919450)

[Figure 4:Preprocessor 11](#_Toc66919451)

[Figure 5:main() function 11](#_Toc66919452)

[Figure 6: All target 12](#_Toc66919453)

[Figure 7: Executable generation and running 12](#_Toc66919454)

[Figure 8: output 12](#_Toc66919455)

[Figure 9: creating new file 13](#_Toc66919456)

[Figure 10: 9 lines commited 13](#_Toc66919457)

[Figure 11: INC and SRC folder 15](#_Toc66919459)

[Figure 12: including the files 15](#_Toc66919460)

[Figure 13: function Calling 16](#_Toc66919461)

[Figure 14: V model 18](#_Toc66919465)

[Figure 15: functionalities of calculator 21](#_Toc66919466)

[Figure 16: distribution of tasks 22](#_Toc66919467)

[Figure 17: SRC and INC 23](#_Toc66919468)

[Figure 18: makefile 23](#_Toc66919469)

[Figure 19: High level test plan 24](#_Toc66919471)

[Figure 20: low level test cases 1 25](#_Toc66919472)

[Figure 21: low level test cases 2 25](#_Toc66919473)

[Figure 22: low level test cases 3&4 25](#_Toc66919474)

[Figure 23: Badges 26](#_Toc66919475)

[Figure 24: Output 27](#_Toc66919477)

[Figure 25: workflows 27](#_Toc66919478)

# SDLC: Software Development Life Cycle

SDLC stands for Software Development Life Cycle. It is a process that gives a complete idea about developing, designing, and maintaining a software project by ensuring that all the functionalities along with user requirements, objectives, and end goals are addressed.

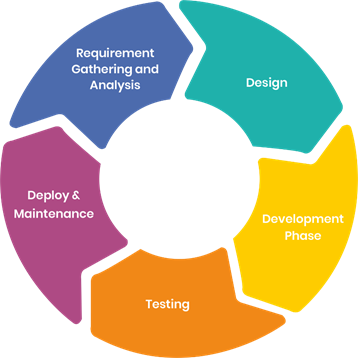


Figure 1:SDLC

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

Steps involved:

1. **Requirement:**

Gathering all possible requirement of Software to be developed such that all customer requirements are satisfied.

1. **System Design:**

The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

1. **Implementation/Execution:**

 With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

1. **Integration:**

All the units developed in the implementation phase are integrated into a system after testing of each unit.

1. **Deployment/Release:**

After satisfying all the requirement of user finalized software version is deployed in the market.

1. **Maintenance**

Any issues in the production environment are resolved by the developers which come under maintenance

## SDLC models:

There are several models used for SDLC:

1. Waterfall Model
2. V-Model
3. Iterative Model
4. Spiral Model
5. Agile Model
6. **Waterfall model:**

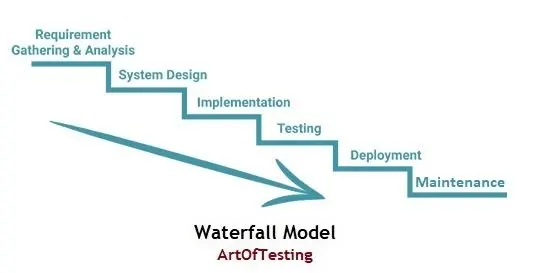


Figure 2:waterfall model

The Waterfall Model was first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**.  It is very simple to understand and use.  In a waterfall model, each phase must be completed fully before the next phase can begin.

* **Advantages and Disadvantages of Waterfall Model:**

**Advantages**

* This model is simple and easy to understand and use.
* It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
* In this model phases are processed and completed one at a time. Phases do not overlap.
* Waterfall model works well for smaller projects where requirements are clearly defined and very well understood.

**Disadvantages**

* Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
* No working software is produced until late during the life cycle.
* High amounts of risk and uncertainty.
* Not a good model for complex and object-oriented projects.
* Poor model for long and ongoing projects.
* Not suitable for the projects where requirements are at a moderate to high risk of changing

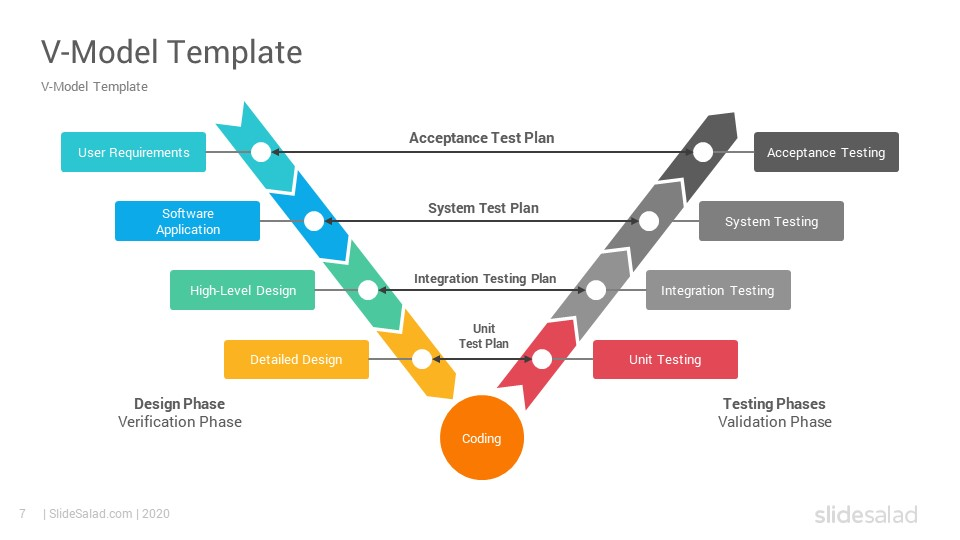
1. **V model:** 

Figure 2: V model

V- model means Verification and Validation model. Just like the waterfall model, the V-Shaped life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins**. V-Model** is one of the many software development models. Testing of the product is planned in parallel with a corresponding phase of development in V-model.

* **Advantages and Disadvantages of V Model:**

**Advantages**

* Simple and easy to use.
* Testing activities like planning, [test designing](http://tryqa.com/what-is-test-design-or-how-to-specify-test-cases/) happens well before coding. This saves a lot of time. Hence higher chance of success over the waterfall model.
* Proactive defect tracking – that is defects are found at early stage.
* Avoids the downward flow of the defects.
* Works well for small projects where requirements are easily understood.

**Disadvantages**

* Very rigid and least flexible.
* Software is developed during the implementation phase, so no early prototypes of the software are produced.
* If any changes happen in midway, then the test documents along with requirement documents must be updated.

1. **Iterative model:**

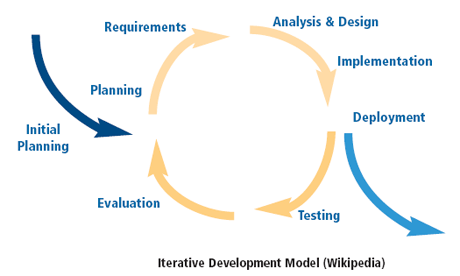


Figure 3: Iterative model

An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which can then be reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software for each cycle of the model.

* **Advantages and Disadvantages of Waterfall Model:**

**Advantages**

* Each phase of an iteration is rigid with no overlaps
* Costly system architecture or design issues may arise because not all requirements are gathered up front for the entire lifecycle.

**Disadvantages**

* Requirements of the complete system are clearly defined and understood.
* When the project is big.
* Major requirements must be defined; however, some details can evolve with time.
  1. **Spiral model:**

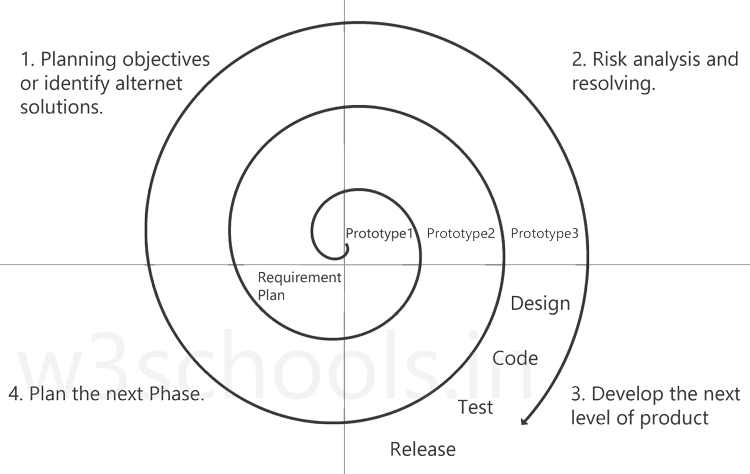


Figure 2:waterfall model

The spiral model is similar to the [incremental model](http://tryqa.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/), with more emphasis placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations (called Spirals in this model). The baseline spiral, starting in the planning phase, requirements are gathered, and risk is assessed.

* **Advantages and Disadvantages of Spiral Model:**

**Advantages**

* High amount of risk analysis hence, avoidance of Risk is enhanced.
* Good for large and mission-critical projects.
* Additional Functionality can be added later.
* Software is produced early in the **software life cycle**.

**Disadvantages**

* Can be a costly model to use.
* Risk analysis requires highly specific expertise.
* Project’s success is highly dependent on the risk analysis phase.
* Does not work well for smaller projects.

1. **Agile Model:**

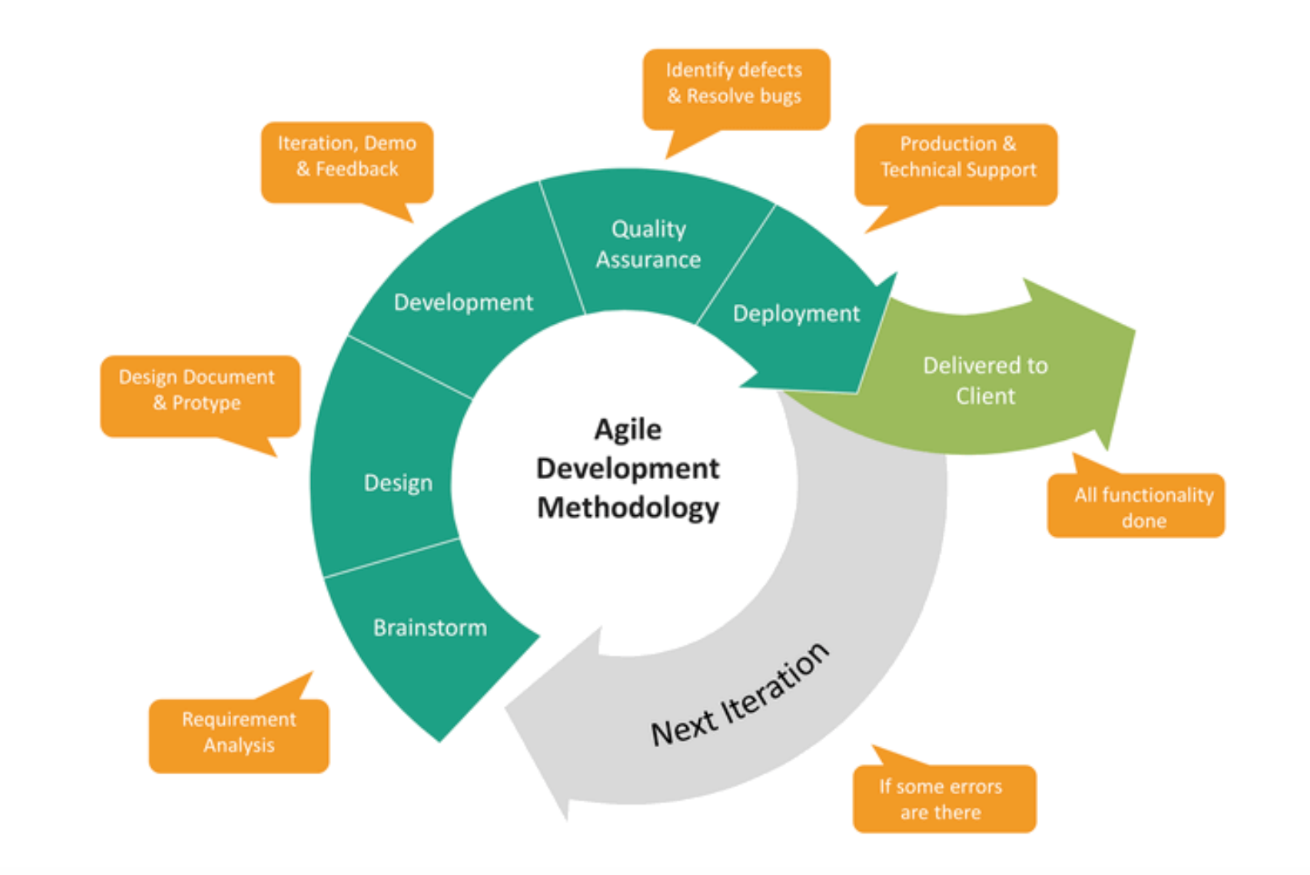


Figure 3:Agile Model

Agile development model is also a type of Incremental model. Software is developed in incremental, rapid cycles. This results in small incremental releases with each release building on previous functionality. Each release is thoroughly tested to ensure software quality is maintained. It is used for time critical applications.

Following are the Agile Manifesto principles −

* **Individuals and interactions** − In Agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.
* **Working software** − Demo working software is considered the best means of communication with the customers to understand their requirements, instead of just depending on documentation.
* **Customer collaboration** − As the requirements cannot be gathered completely in the beginning of the project due to various factors, continuous customer interaction is very important to get proper product requirements.
* **Responding to change** − Agile Development is focused on quick responses to change and continuous development.
* **Advantages and Disadvantages of Spiral Model:**

**Advantages**

* Customer satisfaction by rapid, continuous delivery of useful software.
* Customers, developers and testers constantly interact with each other.
* Working software is delivered frequently (weeks rather than months).
* Close, daily cooperation between businesspeople and developers.
* Continuous attention to technical excellence and good design.
* Regular adaptation to changing circumstances.
* Even late changes in requirements are welcomed

**Disadvantages**

* In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.
* There is lack of emphasis on necessary designing and documentation.
* The project can easily get taken off track if the customer representative is not clear what outcome that they want.
* Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it has no place for newbie programmers, unless combined with experienced resources.

# ACTIVITES

## Activity 1:

* Task: “develop C source file using VSCode text editor and execute the file with Makefile”
* Requirement**:**

Text editor and compiler specifications are mentioned below.

|  |  |  |
| --- | --- | --- |
| Sr. no. | Name | Version |
| 1. | VSCode text editor | Version :1.53  Link: https://code.visualstudio.com/updates/v1.53 |
| 2. | GCC compiler | Version: gcc (MinGW.org GCC-6.3.0-1) 6.3.0  Link: https://sourceforge.net/projects/tdm-gcc/ |
| 3. | Cmake | Link: http://gnuwin32.sourceforge.net/packages/make.htm |

* Explanation**:**

1. Step1:

Installing VSCode, GCC compiler, Cmake in system to create the executable environment.

1. Step2:

Creating the “.C” file using VScode.

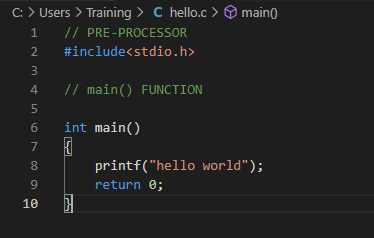


Figure 5: file created as hello.

1. Step3:

Writing the Makefile,



Figure 6: All target

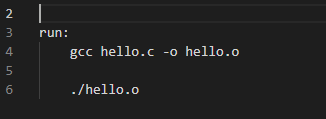


Figure 7: Executable generation and running

1. Result:

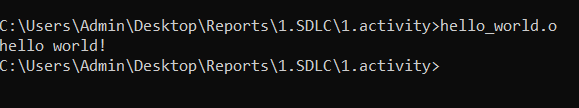


Figure 8: output

## Activity 2a:

* Task**:** “individual commits in same file in GitHub.”

## Explanation:

1. Create one source file and push it to GitHub
2. Individual commit in same file.
3. Execute the file.

**Step1**: Create file in GitHub.

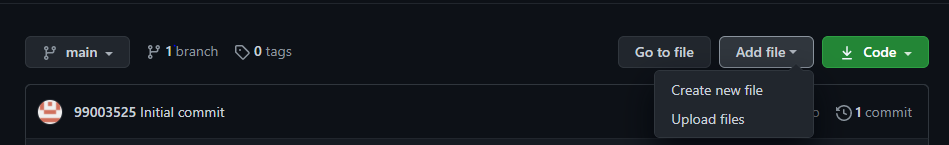


Figure 9: creating new file

**Step2**: Committing the lines to file.

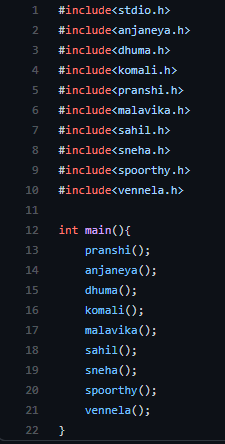


Figure 10: 9 lines committed

**Step3**: Execution of file.

## Activity 2b:

* Task: “using separate header and source files in a group generate executable file and execute the file.”
* Explanation**:**

1. Separate each line of code as a source file.
2. Create separate file in separate INC and SRC folder
3. Generate executable using Make file and execute.
4. Step1:

Create and generate the separate source files,

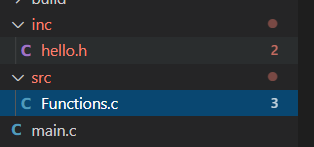


Figure 11: INC and SRC folder

1. Step2:

Including the files

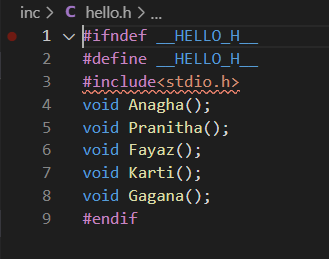


Figure 12: including the files

1. Step3:

Calling the functions,

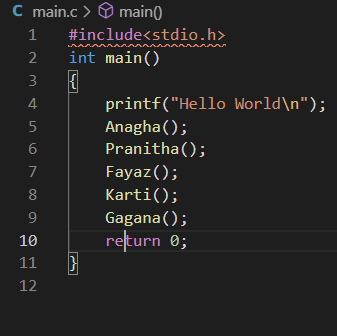


Figure 13: function Calling

## Activity 2c:

* Task**:** “develop the calculator program using the process of SDLC.

# REQUIREMENTS

## BRIEF HISTORY OF A CALCULATOR

People's best friend has always been, and will always be, calculator. This powerful yet diminutive device has undergone a few significant facelifts over the millennia but their basic functions would be familiar in concept to our ancestors. In the very beginning the abacus was introduced. It is a sort of hand operated mechanical calculator using beads on rods, first used by Sumerians and Egyptians around 2000 B C. The abacus was a table of successive columns with beads or stones representing a single unit, which could be used for addition or subtraction. Fast forward 4,500 years to 1617, when Scottish mathematician John Napier published Radiology or “calculation with rods.” In 1642, the first true “calculator” was invented: one that performed calculations through a clockwork-type of mechanism. The Pascal calculator, invented by French inventor and mathematician Blaise Pascal, was lauded for attempting arithmetic calculations previously thought impossible. Curt Herzstark invented the first handheld, mechanical calculator in 1945 from a design he had created in 1938.In 1948 the pocket-sized calculator was invented and came in the form of the Curta calculator. The first solid-state electronic calculator was created in the early 1960s. Pocket-sized devices became available in the 1970.In 1990 TI-81 the first graphics calculator developed for math education that added a new visual dimension to mathematics instruction. In the mid-2000s many of the modern smart phones were introduced and with them came a new way of calculation. Modern scientific calculators are used nowadays for all the purpose. Calculators are now in every phone, every tablet, and in every app store from 1$ to 600$.

X axis – Year Y axis – Cost( percent)

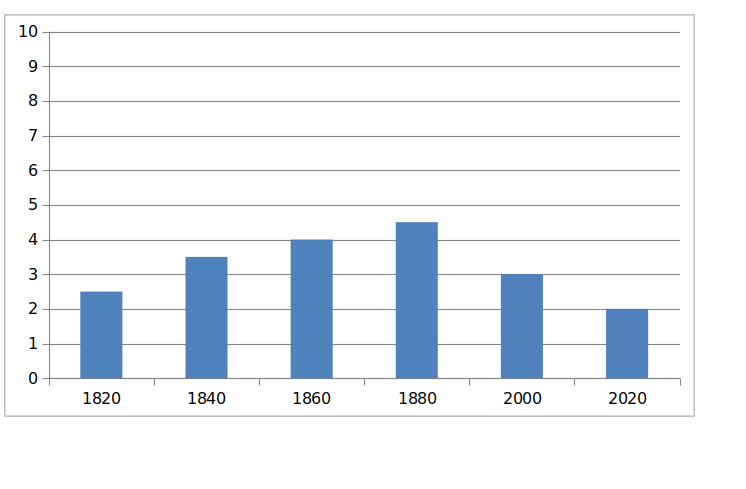
[](https://github.com/99003518/Team2_calciapp/blob/main/Calculator%20Application/1.Requirements/ageandcost.png)

Figure 16: Year vs Cost

### 

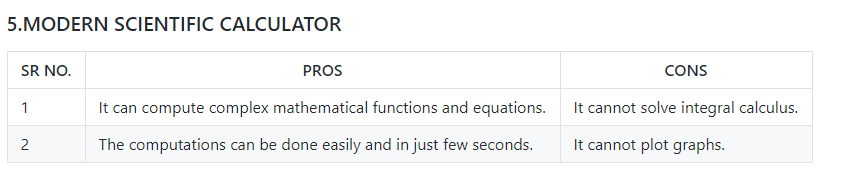


Figure 17: Feature Description.

## 

## PRODUCT DEFINITION

### 1(A). OLD FEATURES OF THE DEVICE

-An electronic calculator is typically a portable electronic device used to perform calculations, ranging from basic arithmetic to complex mathematics. Detailed requirements Low level – percentage is also referred as percent and it is used to express the fraction of a number out of 100%. Percentage means per one hundred, for example 45% represents 45 out of 100 or 45 percent of the total.

-The first solid-state electronic calculator was created in the early 1960s. Pocket-sized devices became available in the 1970s, especially after the Intel 4004, the first microprocessor, was developed by Intel for the Japanese calculator company Busicom. They later became used commonly within the petroleum industry (oil and gas).

-Modern electronic calculators vary from cheap, give-away, credit-card-sized models to sturdy desktop models with built-in printers. They became popular in the mid-1970s as the incorporation of integrated circuits reduced their size and cost. By the end of that decade, prices had dropped to the point where a basic calculator was affordable to most and they became common in schools.

-Computer operating systems as far back as early Unix have included interactive calculator programs such as dc and hoc, and calculator functions are included in almost all personal digital assistant (PDA) type devices, the exceptions being a few dedicated address book and dictionary devices.

-In addition to general purpose calculators, there are those designed for specific markets. For example, there are scientific calculators which include trigonometric and statistical calculations. Some calculators even have the ability to do computer algebra. Graphing calculators can be used to graph functions defined on the real line, or higher-dimensional Euclidean space. As of 2016, basic calculators cost little, but scientific and graphing models tend to cost more.

-In 1986, calculators still represented an estimated 41% of the world's general-purpose hardware capacity to compute information. By 2007, this had diminished to less than 0.05%

### 1(B). FUNCTIONS THAT CAN PERFORMED FROM THE DEVICE

MC or CM Memory Clear

MR, RM, or MRC Memory Recall

M− Memory Subtraction

M+ Memory Addition

C or AC All Clear

CE Clear (last) Entry; sometimes called CE/C: a first press clears the last entry (CE), a second press clears all (C)

± or CHS Toggle positive/negative number aka CHange Sign

% Percent

÷ Division

× Multiplication

− Subtraction

+ Addition

. Decimal point

√ Square root

= Result

### 1(C). FUNCTIONS THAT WE WILL INCLUDE IN OUR DEVICE

% Percent

÷ Division

× Multiplication

− Subtraction

+ Addition

. Decimal point

√ Square root

= Result

p&c Permutations and Combinations

conversions kilometres to meters

sin,cos,tan trigonometric functions

! factorial

e,e^ exponential

[] modulo

|| bitwise operators43.18

### 1(D). TARGET AUDIENCE

Our target audience are students specifically engineering students, CA's, professionals and any other individual who would require to perform such operations at a single stop.

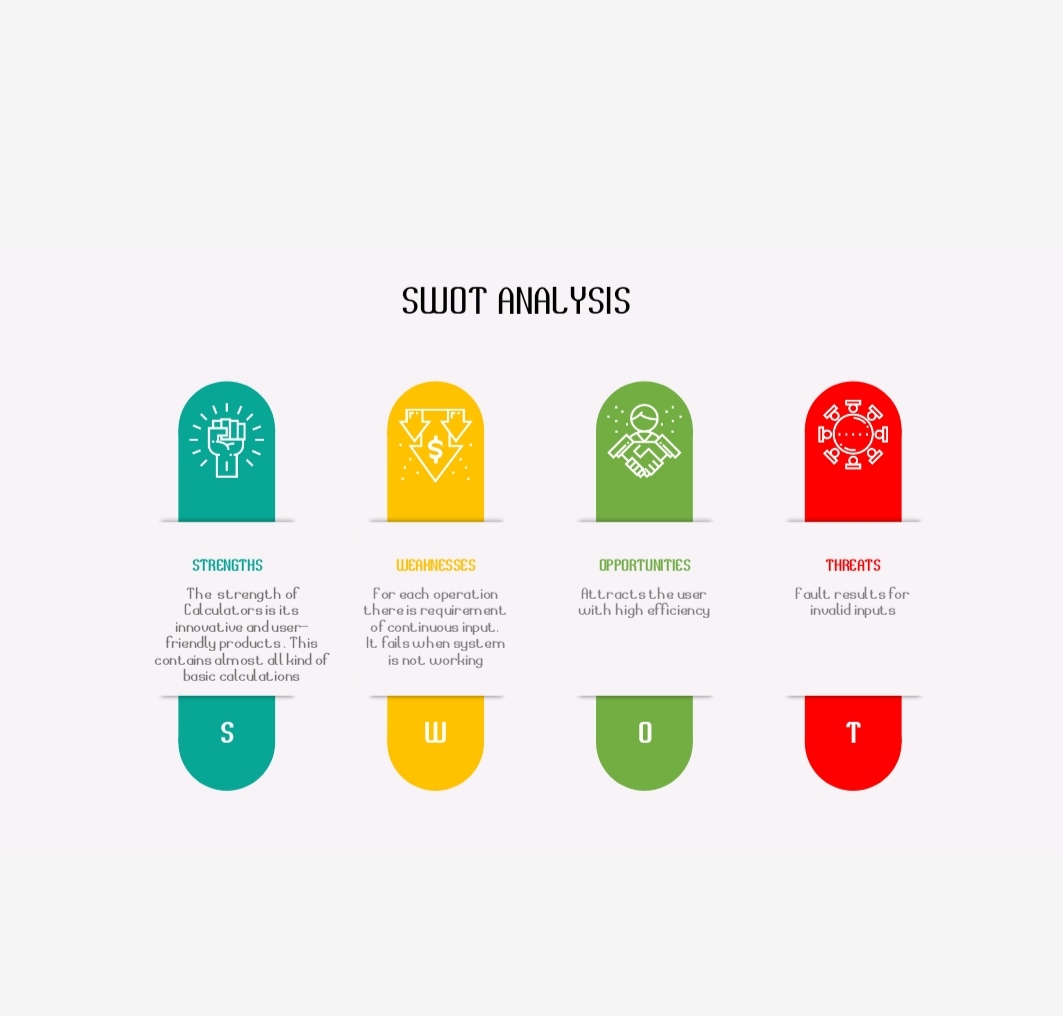
[](https://github.com/enskomali/pccalci/blob/main/Requirements/WhatsApp%20Image%202021-02-05%20at%201.00.35%20PM.jpeg)

Figure 19: SWOT Analysis

## 4W-1H

### What

A calculator is a small hand-held computer that performs mathematical operations. Some calculators even permit simple text editing and programming.

### When

Usually people use calculators in everyday life to save their time and to get accurate answers while doing calculations.

### Where

Many times, in scientific and mathematical calculations involving complex operations with complicated numbers, the calculations are not feasible because it will take a lot of time and there are many chances of getting errors when done manually.

### Why

To make calculations easier.

### How

The memory chips inside the calculator contain thousands or millions of bytes program code that allows the calculator to do work.

### Valid inputs for our device are

It can take up to 2 inputs at a time.

## HIGH LEVEL AND LOW LEVEL REQUIREMENTS

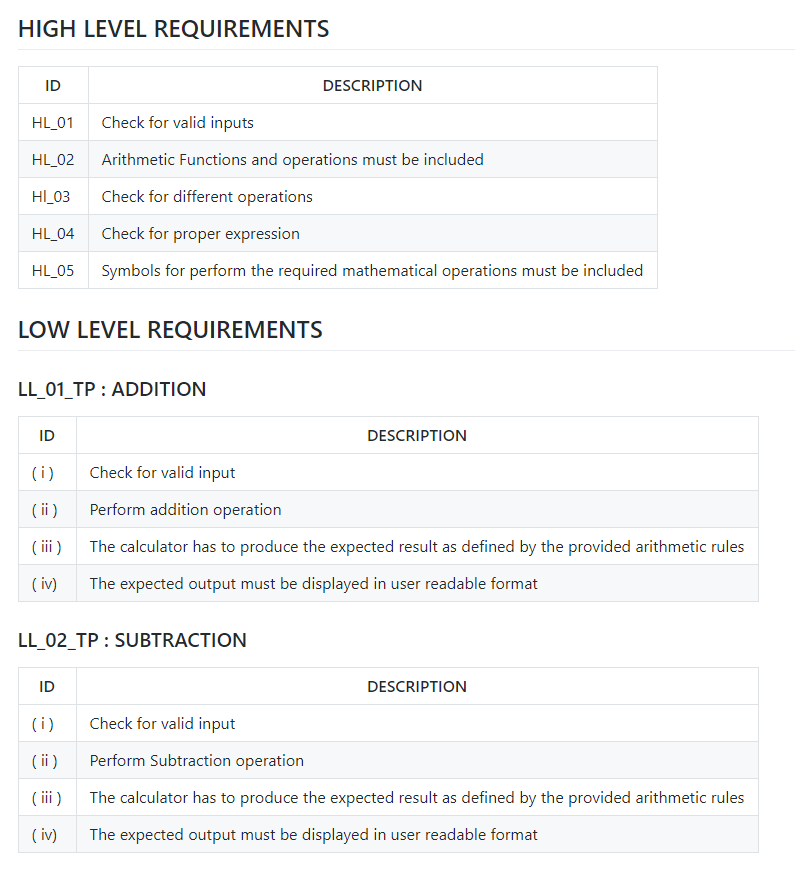


Figure 20: Requirements.

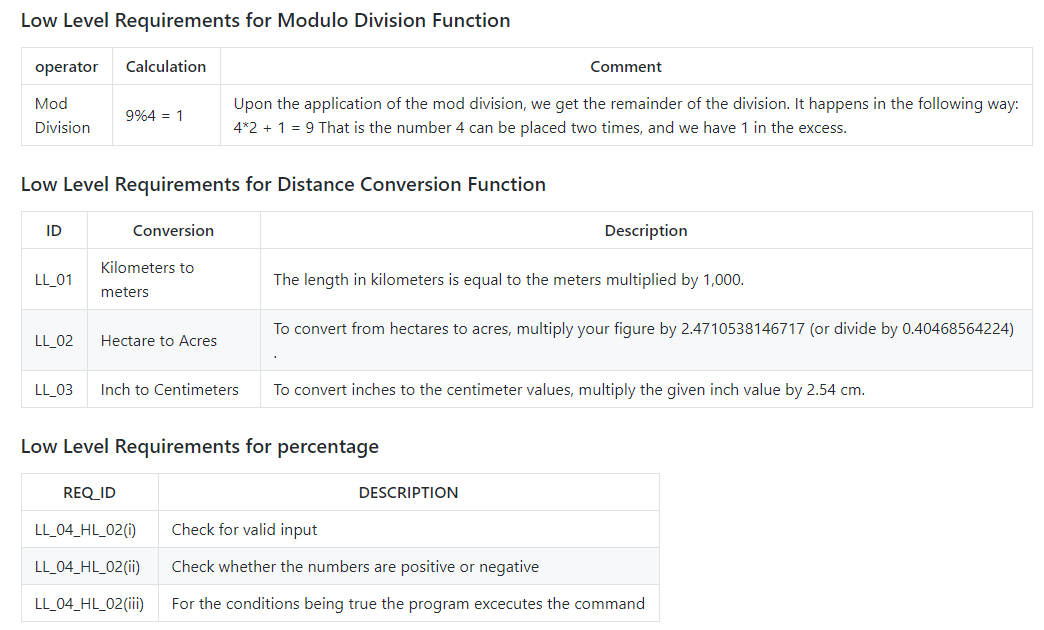


Figure 21: Requirements.

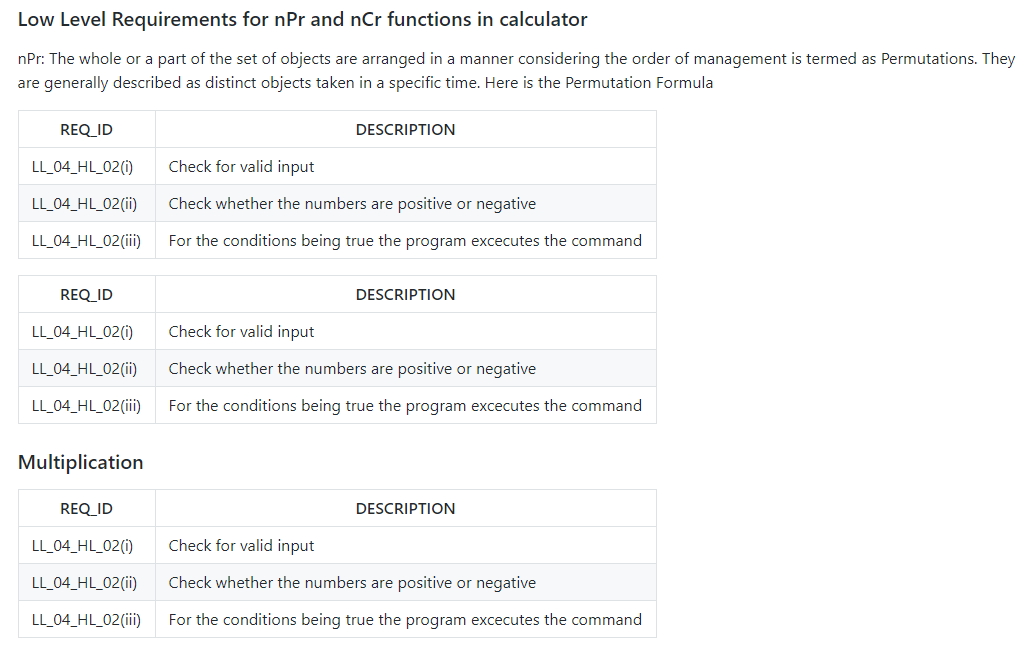


Figure 22: Requirements.

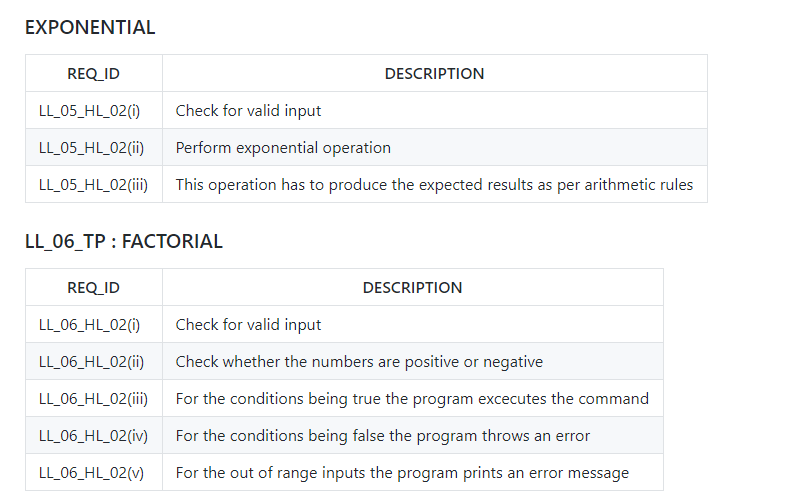


Figure 23: Requirements.

### Procedure to be followed if multiple functions are selected together

The operation that is selected at the first would be performed in the first place.

# High Level design

## Structural diagram

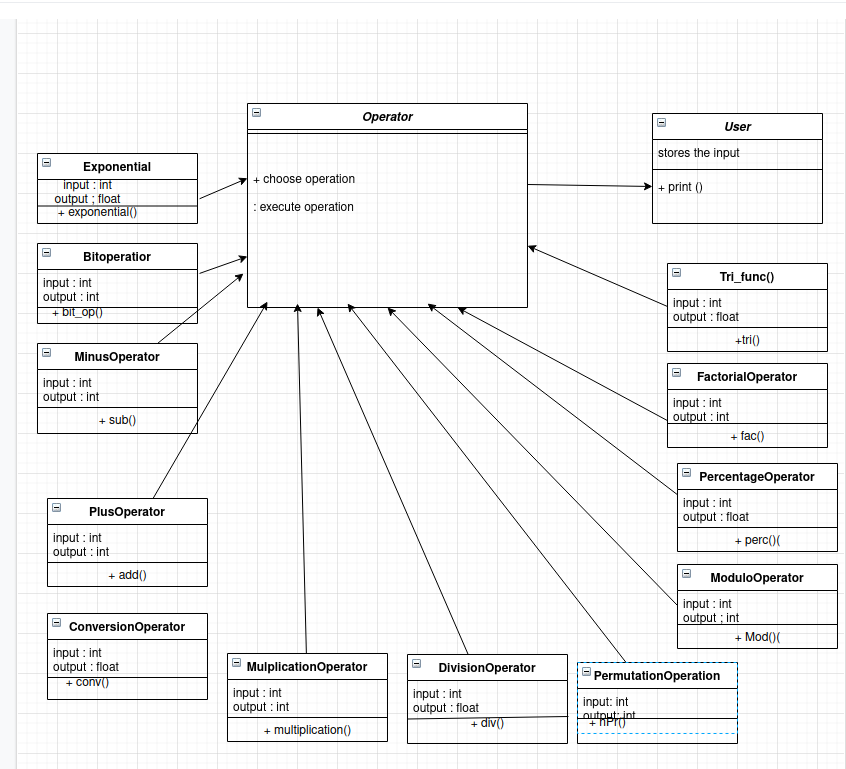


Figure 24: HLD Structural diagram.

### Behavioural Diagram

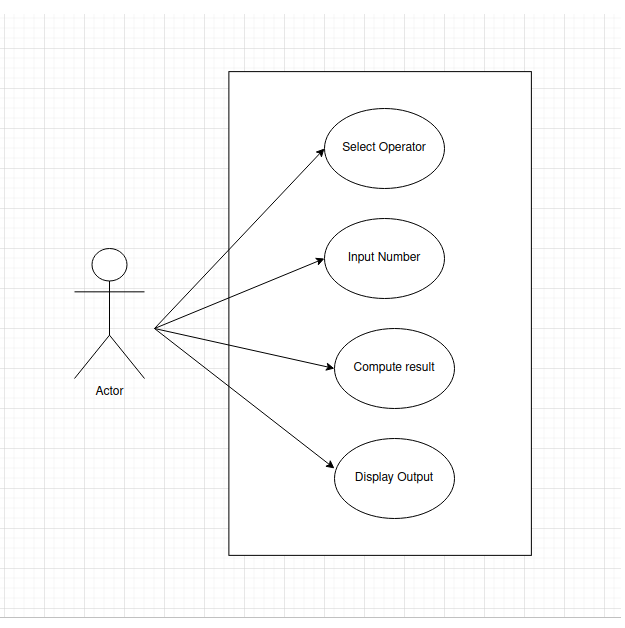


Figure 25: HLD Behavioral Diagram

# LOW LEVEL DESIGN

## Structural Factorial

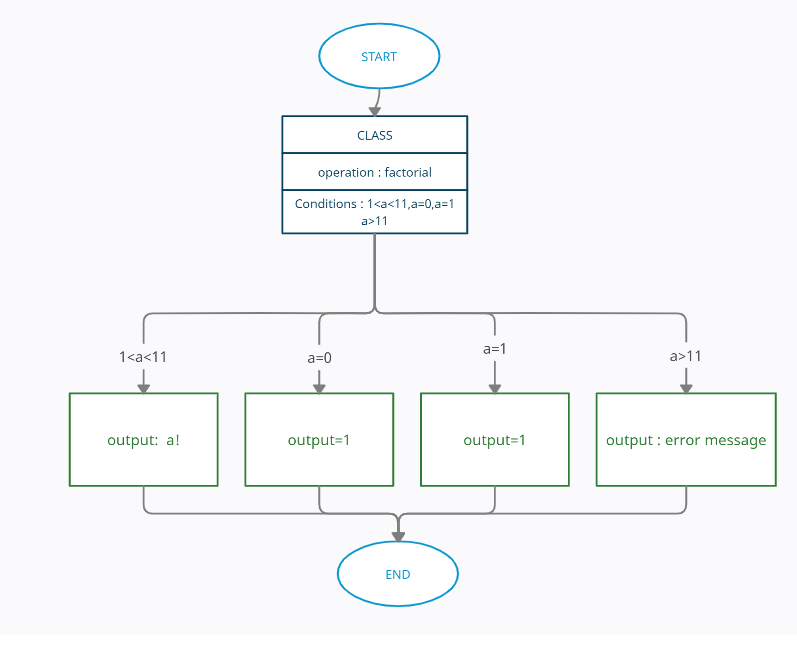


Figure 26: LLD Structural Diagram.

## Behavioural Factorial

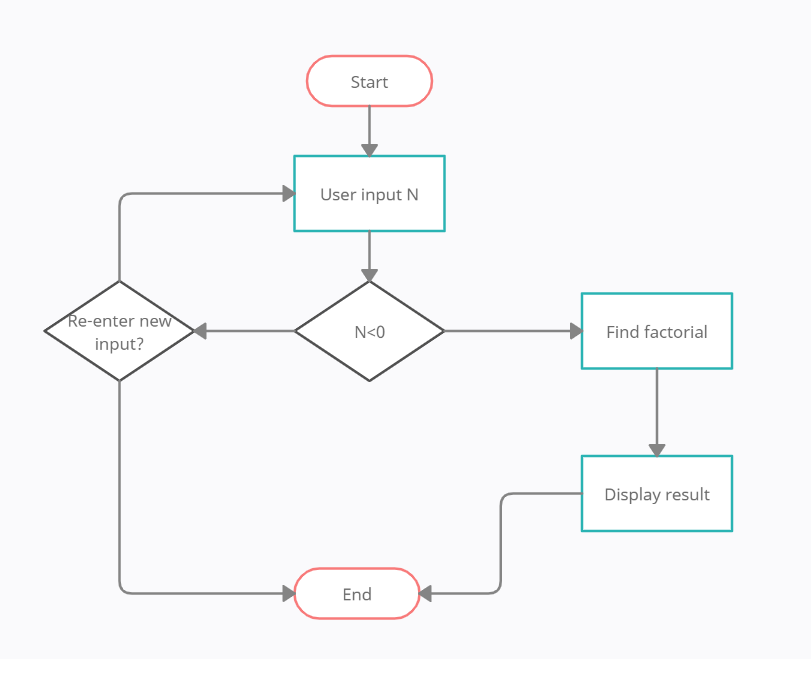


Figure 27: LLD Behavioral Diagram.

# High Level Test Plan



Figure 28: High Level Test Plan.

## LOW LEVEL TEST PLAN

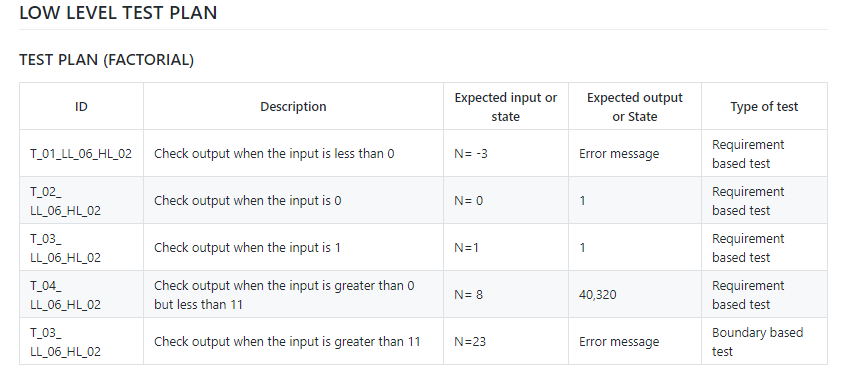


Figure 29: Low Level Test Plan.

### SCOPE OF TESTING

The scope of a test defines what areas of a customer's product are supposed to get tested, what functionalities to focus on, what bug types the customer is interested in, and what areas or features should not be tested by any means. If something is in scope, please test it; if something is out of scope, it should not be tested.

Memory Management cannot be introduced in this device. Multiple inputs are not supposed to be given. Complex numbers are not supported.

# Implementation

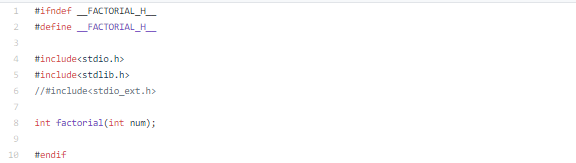


Figure 30: Header File.

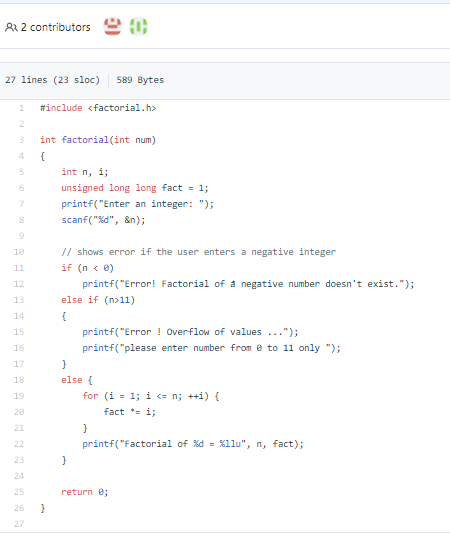


Figure 31: Source File.

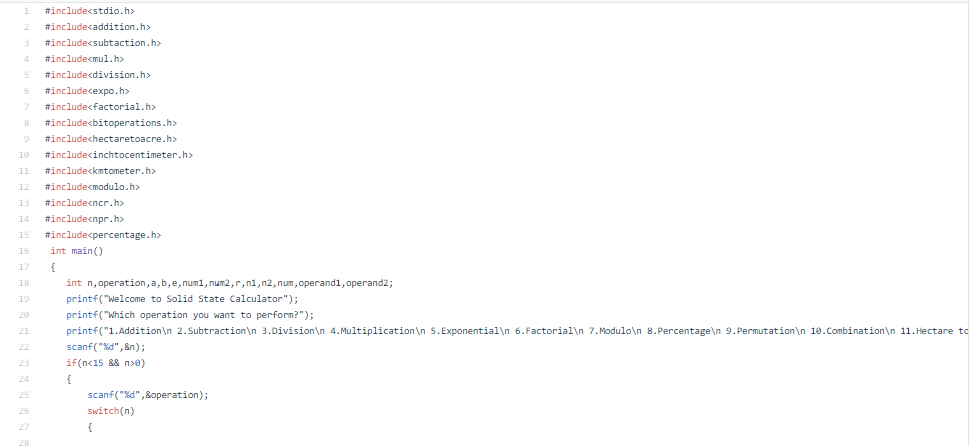
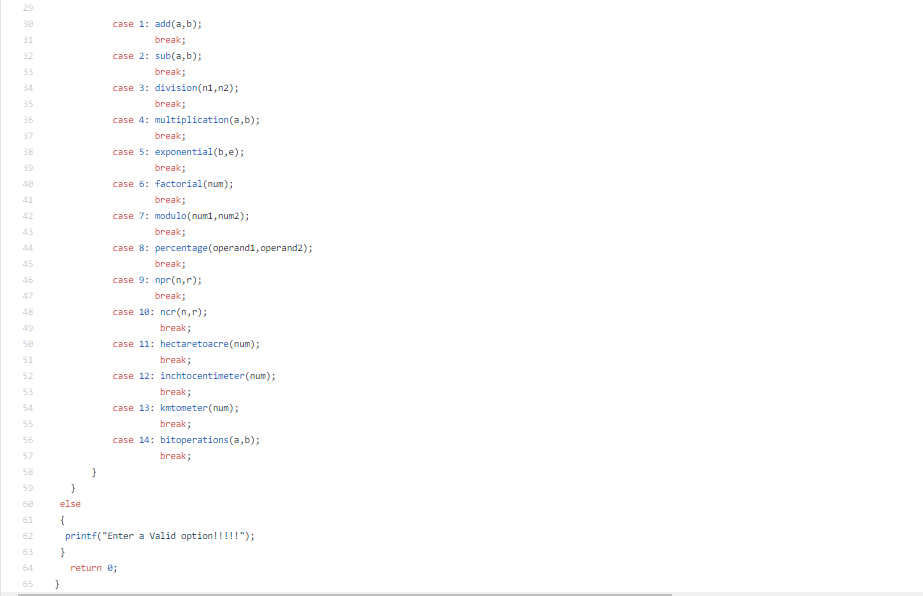
 

Figure 32: Main File.

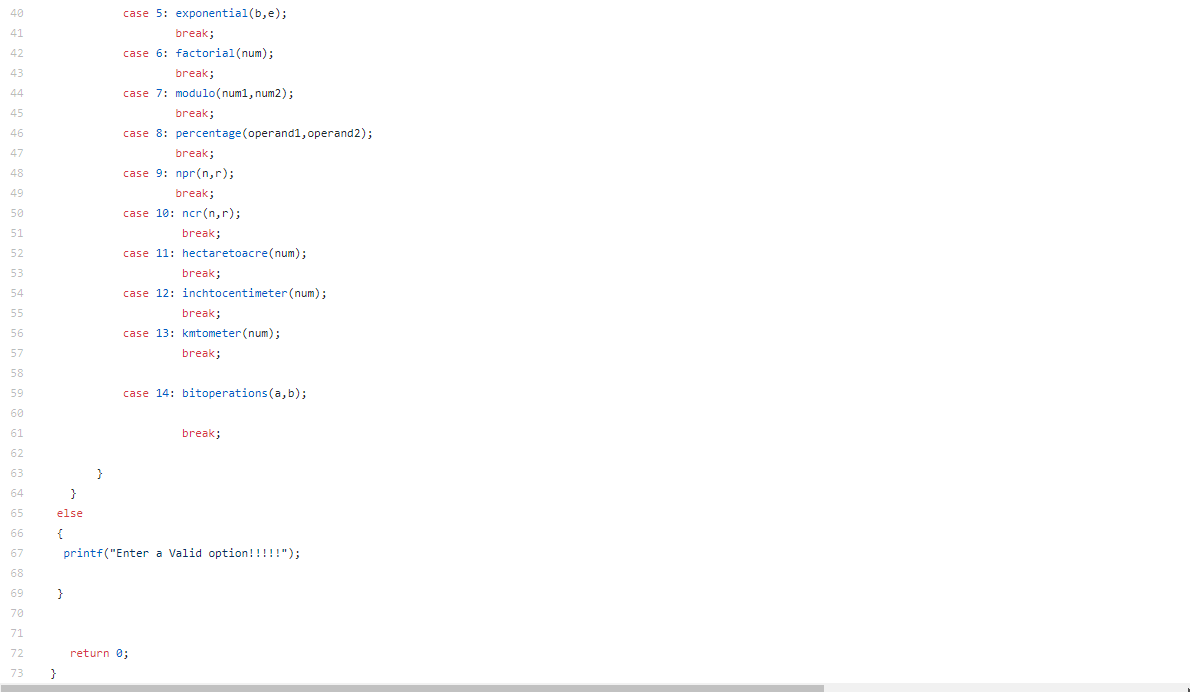
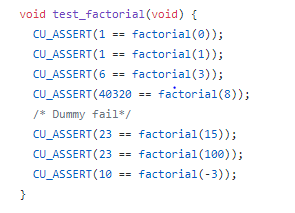


Figure 33: Main File.



|  |
| --- |
| Figure 34: Test File. |
| Figure 35: MakeFile. |  |
|  |  |
|  |  |
| Figure 36: MakeFile. |  |
| GitHub Repositories Activity 1: <https://github.com/stepin105110/test>  Activity 2(a) : <https://github.com/stepin105110/group2>  Activity 2(b) : <https://github.com/stepin105110/group2>  Activity 2(c) : <https://github.com/99003518/Team2_calciapp> |  |