**ASSIGNMENT NO.**

**Title:** Web tool fo**r** Booth's multiplication algorithm

**Aim :**Implementation and Modeling of  Booth's multiplication algorithm.

**Objective :** Modeling , Designing and Testing of  Booth's multiplication algorithm

**Theory :**

Booth's algorithm examines adjacent pairs of [bits](https://en.wikipedia.org/wiki/Bit) of the *N*-bit multiplier *Y* in signed [two's complement](http://complement/) representation, including an implicit bit below the [least significant bit](http://bit/), *y*-1 = 0. For each bit *yi*, for *i* running from 0 to *N*-1, the bits *yi* and *yi*-1 are considered. Where these two bits are equal, the product accumulator *P* is left unchanged. Where *yi* = 0 and *yi*-1 = 1, the multiplicand times 2*i* is added to *P*; and where *y*i = 1 and *y*i-1 = 0, the multiplicand times 2*i* is subtracted from *P*. The final value of *P* is the signed product.

The representations of the multiplicand and product are not specified; typically, these are both also in two's complement representation, like the multiplier, but any number system that supports addition and subtraction will work as well. As stated here, the order of the steps is not determined. Typically, it proceeds from [LSB](http://bit/) to [MSB](http://bit/), starting at *i* = 0; the multiplication by 2*i* is then typically replaced by incremental shifting of the *P* accumulator to the right between steps; low bits can be shifted out, and subsequent additions and subtractions can then be done just on the highest *N* bits of *P*.[[1]](https://en.wikipedia.org/wiki/Booth's_multiplication_algorithm#cite_note-1) There are many variations and optimizations on these details.

The algorithm is often described as converting strings of 1's in the multiplier to a high-order +1 and a low-order –1 at the ends of the string. When a string runs through the MSB, there is no high-order +1, and the net effect is interpretation as a negative of the appropriate value.

Booth's algorithm can be implemented by repeatedly adding (with ordinary unsigned binary addition) one of two predetermined values *A* and *S* to a product *P*, then performing a rightward [arithmetic shift](http://shift/) on *P*. Let **m** and **r** be the multiplicand and multiplier, respectively; and let *x* and *y* represent the number of bits in **m** and **r**.

1. Determine the values of *A* and *S*, and the initial value of *P*. All of these numbers should have a length equal to (*x* + *y* + 1).
   1. A: Fill the most significant (leftmost) bits with the value of **m**. Fill the remaining (*y* + 1) bits with zeros.
   2. S: Fill the most significant bits with the value of (−**m**) in two's complement notation. Fill the remaining (*y* + 1) bits with zeros.
   3. P: Fill the most significant *x* bits with zeros. To the right of this, append the value of **r**. Fill the least significant (rightmost) bit with a zero.
2. Determine the two least significant (rightmost) bits of *P*.
   1. If they are 01, find the value of *P* + *A*. Ignore any overflow.
   2. If they are 10, find the value of *P* + *S*. Ignore any overflow.
   3. If they are 00, do nothing. Use *P* directly in the next step.
   4. If they are 11, do nothing. Use *P* directly in the next step.
3. [Arithmetically shift](http://shift/) the value obtained in the 2nd step by a single place to the right. Let *P* now equal this new value.
4. Repeat steps 2 and 3 until they have been done *y* times.
5. Drop the least significant (rightmost) bit from *P*. This is the product of **m** and **r**.

The functioning of the web server is fairly simple - there is a Python server running that accepts connections from a browser, and serves a basic HTML web page. The server handles a POST request by creating a file and dumping the data in the request to the file. When the length of the file reaches 2, the numbers are added and displayed.

**Software Modelling & designing**

**Mathematical Model:**

Let S be the system such that :

S={s,e,X,Y,F ,Sc,Fc }

Where,

s= initial state

e= end state

X= set of inputs

Y= set of outputs

F= set of function

Sc= Success cases

Fc= Failure cases

Let S’ be system in observation

Where S’ C S

S’ = {s,e,X,Y,F,Sc,Fc}

* S= start state

{init\_arr }

* e= end state

exit(0) ….success

* X= {(i,j) | i,jЄ N}
* Y= {Y1, Y2} Є Y

Where ,

{Y1,}Є success

{ Y2} Є failure

* F= {F1, F2,F3,}

F1 = binary(i)

F2 = rsa()

F3 = con\_decimal(i)

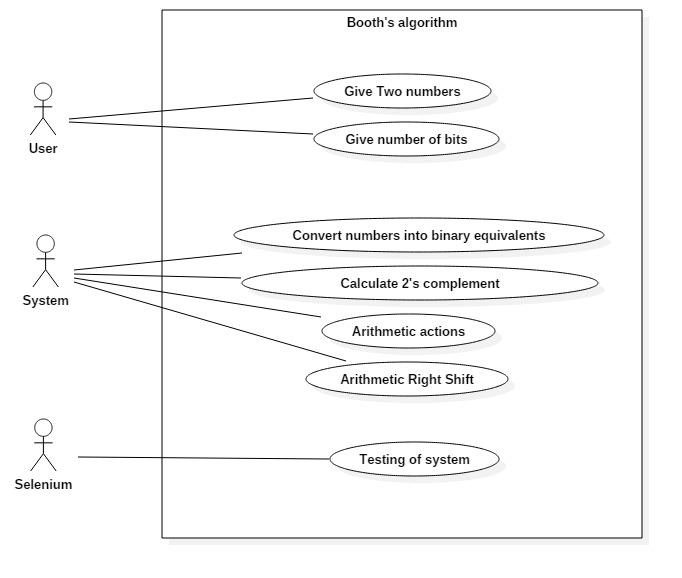
* Sc= {Y1,Y2}

where Y1 = { (i,j) | -7<i,j<7}

* Fc = {Y2}

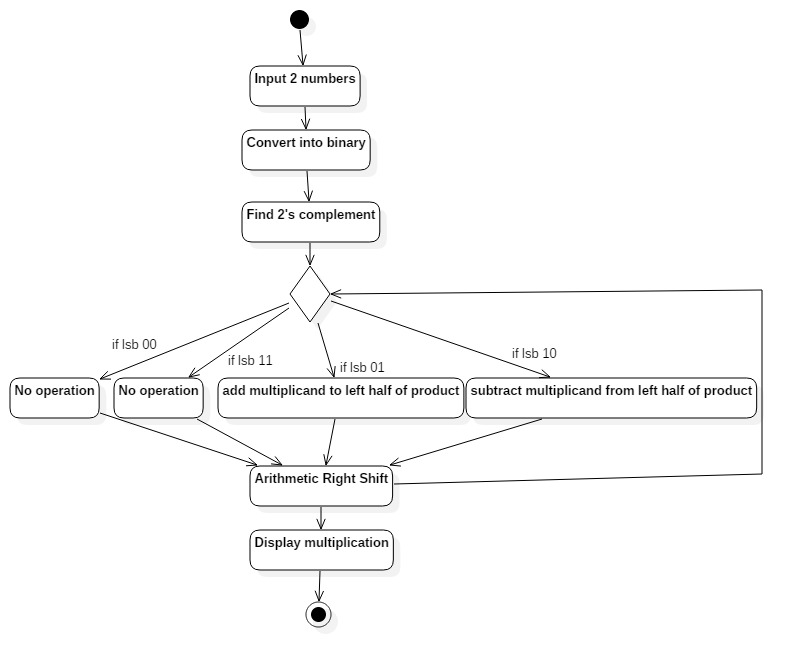
where Y2 = { (i,j) | -7>i,j>7}

**Use Case Diagram:**

****

**Fig: Use Case**

**Activity Diagram:**

****

**Fig: Activity Diagram**

**System Testing:**

Software testing, is based on Manual and Automation Testing. In a project, you can do either Manual or Automation Testing and you can also do both Manual and Automation Testing simultaneously. Manual Testing is a type of Software Testing where Testers manually execute test cases without using any automationtools. Manual testing is the most primitive of all testing types and helps find bugs in the software system.

The key concept of Manual Testing is to ensure that the application is error free and it is working in conformance to the specified functional requirements. It also makes sure that reported defects are fixed by developers and re-testing has been performed by testers on the fixed defects. Basically, this testing checks the quality of the system and delivers bug-free product to the customer.

Automation testing which is also known as “Test Automation”, is when the tester writes scripts and uses another software to test the software. Automation Testing is used to re-run the test scenarios that were performed manually, quickly and repeatedly. Apart from regression testing, Automation testing is also used to test the application from load, performance and stress point of view. It increases the test coverage; improve accuracy, saves time and money in comparison to manual testing.

Following are the tools which can be used for Automation testing:

HP Quick Test Professional

 Selenium

 IBM Rational Functional Tester

 SilkTest

 WinRunner

 LaodRunner

**Selenium**

**Selenium** is a portable [software testing](https://en.wikipedia.org/wiki/Software_testing)[framework](https://en.wikipedia.org/wiki/Software_framework) for [web applications](https://en.wikipedia.org/wiki/Web_application). Selenium provides a record/playback tool for authoring tests without learning a test [scripting language](https://en.wikipedia.org/wiki/Scripting_language) (Selenium IDE). It also provides a test [domain-specific language](https://en.wikipedia.org/wiki/Domain-specific_language) (Selenese) to write tests in a number of popular programming languages, including [Java](https://en.wikipedia.org/wiki/Java_(software_platform)), [C#](https://en.wikipedia.org/wiki/C_Sharp_(programming_language)), [Groovy](https://en.wikipedia.org/wiki/Groovy_(programming_language)), [Perl](https://en.wikipedia.org/wiki/Perl), [PHP](https://en.wikipedia.org/wiki/PHP), [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) and [Ruby](https://en.wikipedia.org/wiki/Ruby_(programming_language)). The tests can then be run against most modern [web browsers](https://en.wikipedia.org/wiki/Web_browser). Selenium deploys on [Windows](https://en.wikipedia.org/wiki/Microsoft_Windows), [Linux](https://en.wikipedia.org/wiki/Linux), and [Macintosh](https://en.wikipedia.org/wiki/Macintosh) platforms. It is [open-source software](https://en.wikipedia.org/wiki/Open-source_software), released under the [Apache 2.0 license](https://en.wikipedia.org/wiki/Apache_License), and can be downloaded and used without charge.

**Selenium IDE** is an integrated development environment for Selenium scripts. It is implemented as a Firefox extension, and allows you to record, edit, and debug tests. Selenium IDE includes the entire Selenium Core, allowing you to easily and quickly record and play back tests in the actual environment that they will run in.

Selenium IDE is not only a recording tool. It is a complete IDE. You can choose to use its recording capability, or you may edit your scripts by hand. With auto complete support and the ability to move commands around quickly, Selenium IDE is the ideal environment for creating Selenium tests no matter what style of tests you prefer.

**Plugins:** As of 1.0.4, Selenium IDE has had a plugin system to allow for easy extension and customization including:

* Adding new functionality to the API
* Changing existing functionality
* Custom formats and export capabilities
* Hosting of plugin update.rdf files
* Adding new locator strategies (coming soon

**Algorithm:**

* Decide which operand will be the **multiplier** and which will be the **multiplicand**
* Convert both operands to **two's complement** representation using X bits.X must be at least one more bit than is required for the binary representation of the numerically larger operand.
* Begin with a product that consists of the multiplier with an additional X leading zero bits.
* Use the **LSB** (least significant bit) and the **previous LSB** to determine the arithmetic action.
* If it is the FIRST pass, use **0** as the previous LSB.
* Possible arithmetic actions:

1. = no arithmetic operation

**11**= add multiplicand to left half of product

**10**= subtract multiplicand from left half of product

**11**= no arithmetic operation

* Perform an **arithmetic right shift** (ASR) on the entire product.

**Input:** Two numbers, number of bits.

**Output:** Multiplication using Booth’s Algorithm.

**Platform:** Ubuntu.

**Language:** jsp and java.

**Conclusion :**Hence, We Implement and designed Booth’s multiplication.