Assignment No	3(Group A-3)
Title	A Web Tool for Booth's multiplication algorithm is used to multiply two numbers located in distributed environment. Use software design client-server architecture and principles for dynamic programming. Perform Risk Analysis. Implement the design using HTML-5/Scala/ Python/Java/C++/ Rubi on Rails. Perform Positive and Negative testing. Use latest open source software modeling, Designing and testing tool/Scrum-it/KADOS and Camel.
Date	
Signature of Faculty	

Assignment No. : 3

1. TITLE

      A Web Tool for Booth's multiplication algorithm is used to multiply two numbers located in distributed environment. Use software design client-server architecture and principles for dynamic programming. Perform Risk Analysis. Implement the design using HTML-5/Scala/ Python/Java/C++/ Rubi on Rails. Perform Positive and Negative testing. Use latest open source software modeling, Designing and testing tool/Scrum-it/KADOS and Camel.

2. PREREQUISITES

- 64-bit Fedora or equivalent OS with 64-bit Intel-i5/i7
- Java 1.7.0
- Testing Tool: Scrum-it/KADOS/Camel

3. OBJECTIVE

- To perform Risk Analysis.
- To learn about designing and testing tools.

4. THEORY

Booth's multiplication algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation. The algorithm was invented by Andrew Donald Booth in 1950.

Booth Multiplication algorithm:
      Booth's algorithm examines adjacent pairs of bits of the *N*-bit multiplier *Y* in signed two's complement representation, including an implicit bit below the least significant bit, $y_{-1} = 0$. For each bit y_i, for *i* running from 0 to *N*-1, the bits y_i and y_{i-1} are considered. Where these two bits are equal, the product accumulator *P* is left unchanged. Where $y_i = 0$ and $y_{i-1} = 1$, 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 multiplicand and product are 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 to MSB, 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

2

out, and subsequent additions and subtractions can then be done just on the highest *N* bits of *P*.
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.

Implementation:
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 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 $(-\mathbf{m})$ 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 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**.

KADOS is a web tool for managing Agile projects (Scrum more specifically) through visual boards on which are displayed post-its representing User Stories, Tasks, Activities, Issues, Actions, Bugs and any objects you wanted your project to manage.

RISK ANALYSIS

Risk can be defined as the potential of losses and rewards resulting from an exposure to a hazard or as a result of a risk event. Risk can be viewed to be a multi-dimensional quantity that includes

• event occurrence probability,
• event occurrence consequences,

3

- consequence significance, and
- the population at risk.

Risk analysis is the process of defining and analyzing the dangers to individuals, businesses and government agencies posed by potential natural and human-caused adverse events. Risk analysis is the review of the risks associated with a particular event or action. In IT, a risk analysis report can be used to align technology-related objectives with a company's business objectives. A risk analysis report can be either quantitative or qualitative. Risk analysis can be defined in many different ways, and much of the definition depends on how risk analysis relates to other concepts. Risk analysis can be "broadly defined to include risk assessment, risk characterization, risk communication, risk management, and policy relating to risk, in the context of risks of concern to individuals, to public- and private-sector organizations, and to society at a local, regional, national, or global level." A useful construct is to divide risk analysis into two components: (1) risk assessment (identifying, evaluating, and measuring the probability and severity of risks) and (2) risk management (deciding what to do about risks).

5. MATHEMATICAL MODELS

Let, S be the System Such that,
A={ S, E, I,O, F, DD, NDD, success, failure }
Where,
S= Start state,
E= End State,
I= Set of Input
O= Set of Out put
F =Set of Function
DD=Deterministic Data
NDD=Non Deterministic Data
Success Case: It is the case when all the inputs are given by system are entered correctly.
Failure Case: It is the case when the input does not match the validation Criteria.

6. CONCLUSION

Hence we have implemented Booth's Multiplication algorithm and performed test cases for that.

7. REFERENCES

[1] HASSAN GOMAA, Software Modeling and Design, Cambridge university Press, 2011, ISBN-13 978-1-107-44735-6**.**
[2] Erich Gamma, Richard Helm, Ralph Johnson ,John Vlissides, Design patterns Elements of Reusable Object-Oriented Software

[3] Srinivasan Desikan, "Software Testing Principals and practices", Pearson Publication ISBN-13 978-8-17-758295-6
[4] https://en.wikipedia.org/wiki/Booth's_multiplication_algorithm.

8. FAQ'S

1. Explain Booth's Multiplication Algorithm.
2. What is Risk Analysis? Give any one example.
3. What is mean by positive and negative testing?
4. Give some examples of positive and negative testing.