ICOER

Computer Laboratory-III
Experiment No. A1

Name:**RAHUL SAWKAR** Roll No: **BA1238**
Branch: Computer Engineering Class: B.E, Sem: II
Date:

Assignment no- A1

Problem Statement:-
Using Divide and Conquer Strategies and object-oriented software design technique using Modelio to design a software function for Binary Search for an un-ordered data stored in memory. Use necessary USE-CASE diagrams and justify its use with the help of mathematical modeling and related efficiency. Implement the design using Eclipse C++ or python.

Algorithm:- Binary search algorithm in C++ relies on a divide and conquer strategy to find a value within an already-sorted collection. Binary search locates the position of an item in a sorted array.

1. Initially Enter data in an unorderd way.
2. Using any sorting algorithm sort the data
3. Binary search compare an input search key to the middle element of the array and the comparison determines whether the element equals the input, less than the input or greater.
4. The return value is the element position in the array.

Binary Search Algorithm complexity:-

- Worst case performance O(log n)
- Best case performance O(1)
- Average case performance O(log n)
- Worst case space complexity O(1)

If you sort your list using quick or merge sort, the complexity becomes O(n*log n). Second part of performing a binary search is done on the 'Sorted list'. The complexity of binary search is O(log n). Therefore ultimately the complexity of the program remains O(n*log n)

Mathematical Model:-

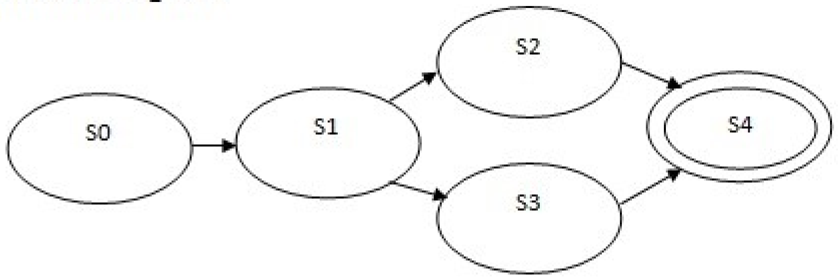
Binary Search

Let S = {Q, ∑, ∂, S0, F}

Where,

Q = Set of States={S0,S1,S2,S3,S4}
∑ = Input (Array of unsorted integer numbers)
∂ = Transition Function
S0 = Initial State
F = Final State

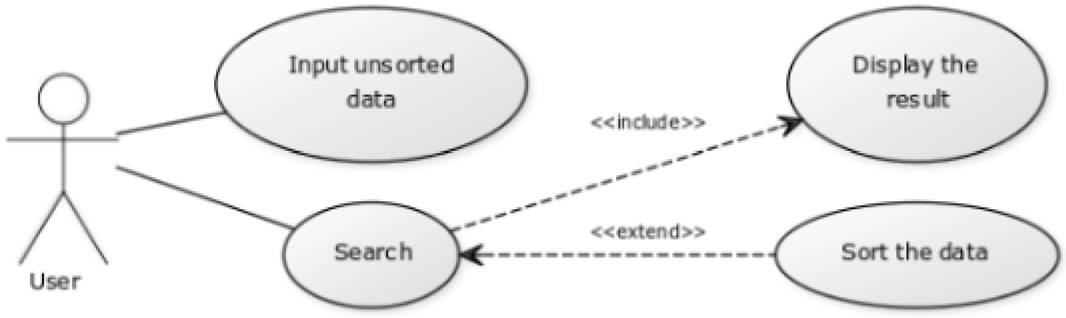
Transition Diagram



Where,

S0= Divide the sorted list by 2
S1= Compare the Key value with midpoint
S2= if key > midpoint perform search in low
S3= if key < midpoint perform search in high
S4= if key found return the position of key else display key not found

Use Case Diagram:-



Theory

In computer science, binary search, also known as half-interval search or logarithmic search, is a search algorithm that finds the position of a target value within a sorted array.It works by comparing the target value to the midpoint of the array; if they are not equal, the lower or upper half of the array is eliminated depending on the result and the search is repeated until the position of the target value is found. Binary search is intuitively recursive; however, it can be done iteratively by keeping track of the bounds of the search with two pointers. Binary search is efficient for sorted arrays that are stored contiguously (close together) in memory, making O(log n) comparisons, where n is the number of elements in the array. The binary search algorithm can be classified as a dichotomic divide-and-conquer search algorithm. Even if we do not know a fixed range the number k falls in, we can still determine its value by asking $2\lceil\log_2 k\rceil$.

For example, suppose we could answer "Does this n x n matrix have permanent larger than k?" in O(n2) time. Then, by using binary search, we could find the (ceiling of the) permanent itself in O(n2 log p) time, where p is the value of the permanent. Notice that p is not the size of the input, but the value of the output; given a matrix whose maximum item (in absolute value) is m, p is bounded by m$^n n!. Hence log p = O(n log n + n log m)$.

If the original number of items is N then after the first iteration there will be at most N/2 items remaining, then at most N/4 items, at most N/8 items, and so on. In the worst case, when the value is not in the list, the algorithm must continue iterating until the span has been made empty; this will have taken at most âŇŁlog2(N)+1âŇŃ iterations, where the âŇŁ âŇŃ notation denotes the floor function that rounds its argument down to an integer. This worst case analysis is tight: for any N there exists a query that takes exactly âŇŁlog2(N)+1âŇŃ iterations. When compared to linear search, whose worst-case behaviour is N iterations, we see that binary search is substantially faster as N grows large. For example, to search a list of one million items takes as many as one million iterations with linear search, but never more than twenty iterations with binary search. However, a binary search can only be performed if the list is in sorted order.

Conclusion:-

Successfully run the binary Search Program using Divide & Conquer Technique for unordered data stored in memory.

ICOER

Computer Laboratory-III

Experiment No. A2

Name:**RAHUL SAWKAR** Roll No: **BA1238**
Branch: Computer Engineering Class: B.E, Sem: II
Date:

Assignment no- A2

Problem Statement:-
Using Divide and Conquer Strategies to design an efficient class for Concurrent Quick Sort and the input data is stored using XML. Use object oriented software design method and Modelio/ StarUML2.x Tool. Perform the efficiency comparison with any two software design methods. Use necessary USE-CASE diagrams and justify its use with the help of mathematical modeling. Implement the design using Scala/Python/Java/C++.

Aim:-
Implement Quick Sort for input data is stored using XML using Scala/Python/Java/C++. Create USE-CASE Diagram using Modelio/ StarUML2.x Tool and justify its use with the help of mathematical modelling and related efficiency.

Algorithm:- Quicksort (sometimes called partition-exchange sort) is comparison sort algorithm, meaning that it can sort items of any type for which a "less-than" operator is defined. In efficient implementations it is not a stable sort, meaning that the relative order of equal sort items is not preserved.
Algorithm

- Quicksort is a divide and conquer algorithm.
- Quicksort first divides a large array into two smaller sub-arrays:
- The low elements and the high elements.

Quicksort can then recursively sort the sub-arrays.
The steps are:
1) Pick an element, called a pivot, from the array.
2) Reorder the array so that all elements with values less than the pivot come before the pivot, while all elements with values greater than the pivot come after it (equal values can go either way).
3) After this partitioning, the pivot is in its final position. This is called the partition operation.
4) Recursively apply the above steps to the sub-array of elements with smaller values and separately to the sub-array of elements with greater values.
5) The base case of the recursion is arrays of size zero or one, which never need to be sorted.

Binary Search Algorithm complexity:-

- Worst case performance O(n2)
- Best case performance O(n log n)
- Average case performance O(n log n)

Quick sort can be easily parallelized due to its divide-and-conquer nature. Individual in-place partition operations are difficult to parallelize, but once divided, different sections of the list can be sorted in parallel. One advantage of parallel quicksort over other parallel sort algorithms is that no synchronization is required. A new thread is started as soon as a sublist is available for it to work on and it does not communicate with other threads. When all threads complete, the sort is done.
So, the computational time would be: O(N/p * log(N/p)), where p is the number of processes. When the number of processes (p) is far below the job size, we know that the computational time is: O(N*log(N/p))

Mathematical Model:-

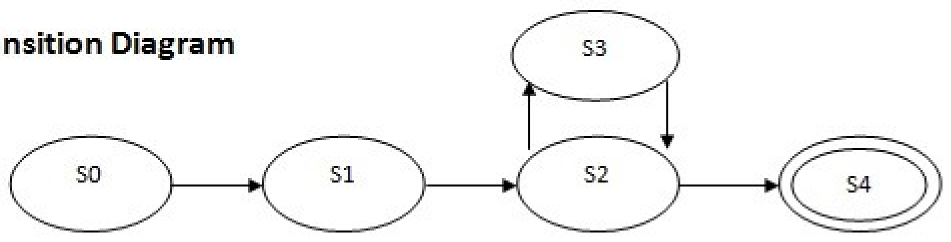
Quick Sort

Let S = {Q, ∑, ∂, S0, F}

Where,

Q = Set of States={S0,S1,S2,S3,S4}
∑ = Input (Array of unsorted integer numbers)
∂ = Transition Function
S0 = Initial State
F = Final State

Transition Diagram



Where,

S0= Input Array of N numbers

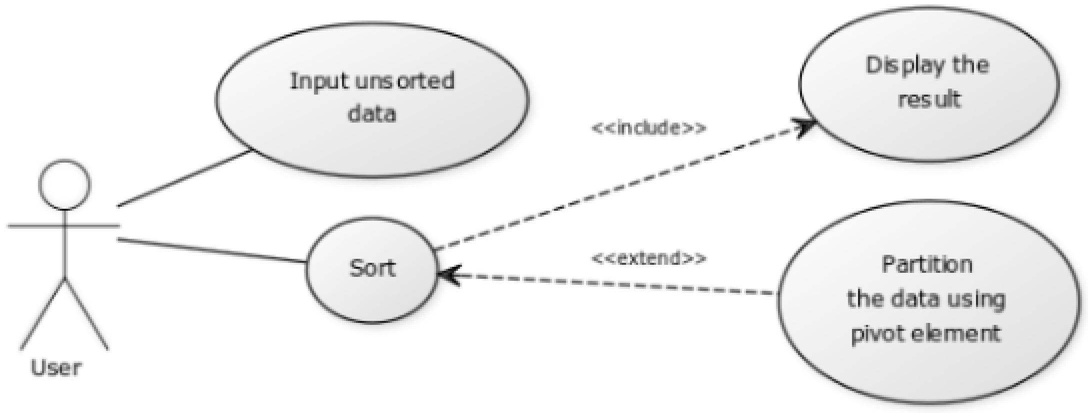
S1= Initialize the Pivot

S2= Reorder the array so that all elements with values less than the pivot come before the pivot, while all elements with values greater than the pivot come after it

S3= Recursively apply the above steps to the sub-array of elements with smaller values and separately to the sub-array of elements with greater values.

S4= Final State (Sorted Array of Numbers)

Use Case Diagram:-



Theory

Quicksort (sometimes called partition-exchange sort) is an efficient sorting algorithm, serving as a systematic method for placing the elements of an array in order. Developed by Tony Hoare in 1959, with his work published in 1961, it is still a commonly used algorithm for sorting. When implemented well, it can be about two or three times faster than its main competitors, merge sort and heapsort.
Quicksort is a comparison sort, meaning that it can sort items of any type for which a "less-than" relation (formally, a total order) is defined. In efficient implementations it is not a stable sort, meaning that the relative order of equal sort items is not preserved. Quicksort can operate in-place on an array, requiring small additional amounts of memory to perform the sorting.

Mathematical analysis of quicksort shows that, on average, the algorithm takes O(n log n) comparisons to sort n items. In the worst case, it makes O(n2) comparisons, though this behavior is rare.

Quicksort's divide-and-conquer formulation makes it amenable to parallelization using task parallelism. The partitioning step is accomplished through the use of a parallel prefix sum algorithm to compute an index for each array element in its section of the partitioned array. Given an array of size n, the partitioning step performs O(n) work in O(log n) time and requires O(n) additional scratch space. After the array has been partitioned, the two partitions can be sorted recursively in parallel. Assuming an ideal choice of pivots, parallel quicksort sorts an array of size n in O(n log n) work in O(logÂš n) time using O(n) additional space.

Quicksort has some disadvantages when compared to alternative sorting algorithms, like merge sort, which complicate its efficient parallelization. The depth of quicksort's divide-and-conquer tree directly impacts the algorithm's scalability, and this depth is highly dependent on the algorithm's choice of pivot. Additionally, it is difficult to parallelize the partitioning step efficiently in-place. The use of scratch space simplifies the partitioning step, but increases the algorithm's memory footprint and constant overheads.

Other more sophisticated parallel sorting algorithms can achieve even better time bounds. For example, in 1991 David Powers described a parallelized quicksort (and a related radix sort) that can operate in O(log n) time on a CRCW PRAM with n processors by performing partitioning implicitly.

Conclusion:-

Successfully run the Quick Sort Program using Divide & Conquer Technique.

ICOER

Computer Laboratory-III
Experiment No. A3

Name:**RAHUL SAWKAR** Roll No: **BA1238**
Branch: Computer Engineering Class: B.E, Sem: II
Date:

Assignment no- A3

Problem Statement:-
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.

Algorithm:- Booths Multiplication:- Let m and r be the multiplicand and multiplier, respectively; and let x and y represent the number of bits in m and r.

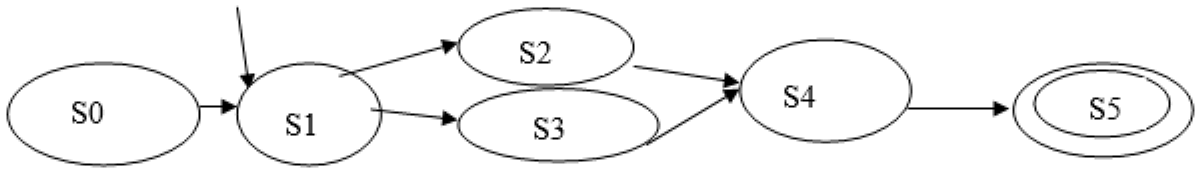
1. Determine the values of A and S, and the initial value of P. All of these numbers should have a length equal to (x + y + 1).
   (a) A: Fill the most significant (leftmost) bits with the value of m. Fill the remaining (y + 1) bits with zeros.
   (b) S: Fill the most significant bits with the value of (-m) in two's complement notation. Fill the remaining (y + 1) bits with zeros.
   (c) 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.
   (a) If they are 01, find the value of P + A. Ignore any overflow.
   (b) If they are 10, find the value of P + S. Ignore any overflow.
   (c) If they are 00, do nothing. Use P directly in the next step.
   (d) 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.

Mathematical Modelling:-

Let S= {Q, Σ, δ, S0, F}

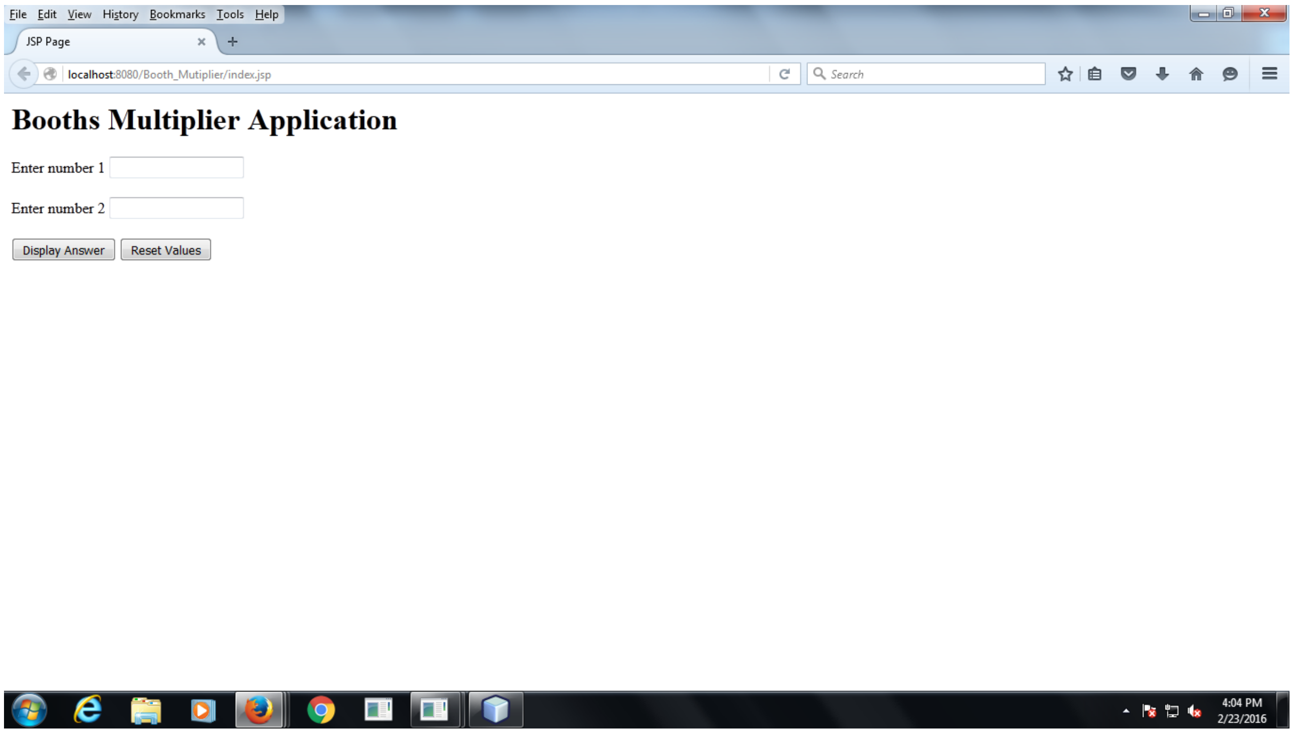
Where,
Q=Set of States= {S0, S1, S2, S3, S4}
Σ=Input (Multiplier & Multiplicand)
δ=Transition Function
S0=Initial State
F= Final State

Transition



Where,
S0= Determine the values of A and S, and the initial value of P
S1= Determine the two least significant (rightmost) bits of P.
S2= If they are 01, find the value of P + A.
S3= If they are 10, find the value of P + S.
S4= arithmetically shift the value obtained in the 2nd step by a single place to the right. Recursively repeat steps from S1 steps until they have been done y times.
S5=Final State (Drop the least significant (rightmost) bit from P)

Risk Analysis:-
1) How likely is it that it will fail?
2) What are the ways it could fail?
3) What can we do in advance to prevent failure?
4) How can we consistently test that this component is healthy?
5) How will we know if this failed?
6) How can we structure this component to be monitor-able through an external system? (A status JSON/XML script generated HTTP status codes, etc - anything you can attach a status monitor to.)
7) How can we structure this component to fail more gracefully? (Firing an alert and redirecting instead of 500 error, for example)


File Edit View History Bookmarks Tools Help
JSP Page
localhost:8080/Booth_Mutiplier/index.jsp
Search
4:04 PM
2/23/2016

Booths Multiplier Application

Enter number 1

Enter number 2

Display Answer Reset Values

Test Cases

Positive cases

Check the calculator if it starts by running the application

Check the if the calculator application closes if the exit is clicked

Check if all the numbers are working (1 to 15) Check if the multiplier key is working.

Check the multiplication of two negative numbers. Check the multiplication of one negative and one positive number.

Check the multiplication of two integer numbers.

Negative cases

Check Numbers which are not between 1 to 15

check the multiplication by entering Multiplicand greater than multiplier

Conclusion:-

Successfully performed Booths multiplier web application to perform multiplication of 2 numbers located in distributed environment.

ICOER

Computer Laboratory-III

Experiment No. A4

Name:**RAHUL SAWKAR** Roll No: **BA1238**
Branch: Computer Engineering Class: B.E, Sem: II
Date:

Assignment no- A4

Problem Statement:-
In an embedded system application Dining Philosopher's problem algorithm is used to design a software that uses shared memory between neighboring processes to consume the data. The Data is generated by different Sensors/WSN system Network and stored in MongoDB (NoSQL). Implementation be done using Scala/ Python/ C++/ Java. Design using Client-Server architecture. Perform Reliability Testing. Use latest open source software modeling, Designing and testing tool/Scrum-it/KADOS, No SQLUnit and Camel..

Learning Objective:
1 Implementation of the problem statement using Object oriented programming.
2 Solve Dining Philosopher's using mongoDB.

Theory: Introduction
In computer science, the dining philosophers problem is an example problem often used in concurrent algorithm design to illustrate synchronization issues and techniques for resolving them.

Five silent philosophers sit at a round table with bowls of spaghetti. Forks are placed between each pair of adjacent philosophers.

Each philosopher must alternately think and eat. However, a philosopher can only eat spaghetti when he has both left and right forks. Each fork can be held by only one philosopher and so a philosopher can use the fork only if it is not being used by another philosopher. After he finishes eating, he needs to put down both forks so they become available to others. A philosopher can take the fork on his right or the one on his left as they become available, but cannot start eating before getting both of them.

Eating is not limited by the remaining amounts of spaghetti or stomach space; an infinite supply and an infinite demand are assumed.

The problem is how to design a discipline of behavior (a concurrent algorithm) such that no philosopher will starve; i.e., each can forever continue to alternate between eating and thinking, assuming that no philosopher can know when others may want to eat or think.

The problem was designed to illustrate the challenges of avoiding deadlock, a system state in which no progress is possible. To see that a proper solution to this problem is not obvious, consider a proposal in which each philosopher is instructed to behave as follows:

- think until the left fork is available;
- when it is, pick it up; think until the right fork is available;
- when it is, pick it up;
- when both forks are held, eat for a fixed amount of time;
- then, put the right fork down;
- then, put the left fork down;
- repeat from the beginning.

This attempted solution fails because it allows the system to reach a deadlock state, in which no progress is possible. This is a state in which each philosopher has picked up the fork to the left, and is waiting for the fork to the right to become available. With the given instructions, this state can be reached, and when it is reached, the philosophers will eternally wait for each other to release a fork.

Resource starvation might also occur independently of deadlock if a particular philosopher is unable to acquire both forks because of a timing problem. For example there might be a rule that the philosophers put down a fork after waiting ten minutes for the other fork to become available and wait a further ten minutes before making their next attempt. This scheme eliminates the possibility of deadlock (the system can always advance to a different state) but still suffers from the problem of livelock. If all five philosophers appear in the dining room at exactly the same time and each picks up the left fork at the same time the philosophers will wait ten minutes until they all put their forks down and then wait a further ten minutes before they all pick them up again.

Mutual exclusion is the basic idea of the problem; the dining philosophers create a generic and abstract scenario useful for explaining issues of this type. The failures these philosophers may experience are analogous to the difficulties that arise in real computer programming when multiple programs need exclusive access to shared resources. These issues are studied in the branch of concurrent programming. The original problems of Dijkstra were related to external devices like tape drives. However, the difficulties exemplified by the dining philosophers problem arise far more often when multiple processes access sets of data that are being updated. Systems such as operating system kernels use thousands of locks and synchronizations that require strict adherence to methods and protocols if such problems as deadlock, starvation, or data corruption are to be avoided.

Resource hierarchy solution

This solution to the problem is the one originally proposed by Dijkstra. It assigns a partial order to the resources (the forks, in this case), and establishes the convention that all resources will be requested in order, and that no two resources unrelated by order will ever be used by a single unit of work at the same time. Here, the resources (forks) will be numbered 1 through 5 and each unit of work (philosopher) will always pick up the lowernumbered fork first, and then the higher-numbered fork, from among the two forks he plans to use. The order in which each philosopher puts down the forks does not matter. In this case, if four of the five philosophers simultaneously pick up their lower-numbered fork, only the highest numbered fork will remain on the table, so the fifth philosopher will not be able to pick up any fork. Moreover, only one philosopher will have access to that highestnumbered fork, so he will be able to eat using two forks.

While the resource hierarchy solution avoids deadlocks, it is not always practical, especially when the list of required resources is not completely known in advance. For example, if a unit of work holds resources

3 and 5 and then determines it needs resource 2, it must release 5, then 3 before acquiring 2, and then it must re-acquire 3 and 5 in that order. Computer programs that access large numbers of database records would not run efficiently if they were required to release all higher-numbered records before accessing a new record, making the method impractical for that purpose.

What is MongoDB?
MongoDB is one of several database types to arise in the mid-2000s under the NoSQL banner. Instead of using tables and rows as in relational databases, MongoDB is built on an architecture of collections and documents. Documents comprise sets of key-value pairs and are the basic unit of data in MongoDB. Collections contain sets of documents and function as the equivalent of relational database tables.

Like other NoSQL databases, MongoDB supports dynamic schema design, allowing the documents in a collection to have different fields and structures. The database uses a document storage and data interchange format called BSON, which provides a binary representation of JSON-like documents. Automatic sharding enables data in a collection to be distributed across multiple systems for horizontal scalability as data volumes increase.

Program Flow:-

- First of all create a server in mongoDB (Use terminal)
- Now Write main program add the mongo jar file.
- Now run the main code to see the output from other terminal using commands like "show databases;" and "db.collection_name.find();"

Reliability Testing:- Software reliability testing a testing technique that relates to testing a software's ability to function given environmental conditions consistently that helps uncover issues in the software design and functionality.
Parameters involved in Reliability Testing:

- Dependent elements of reliability Testing:
- Probability of failure-free operation
- Length of time of failure-free operation The environment in which it is executed

Key Parameters that are measured as part of reliability are given below:

- MTTF: Mean Time To Failure
- MTTR: Mean Time To Repair
- MTBF: Mean Time Between Failures (= MTTF + MTTR)

Reliability refers to the consistency of a measure. A test is considered reliable if we get the same result repeatedly. Software Reliability is the probability of failure-free software operation for a specified period of time in a specified environment. Software Reliability is also an important factor affecting system reliability.

Reliability testing will tend to uncover earlier those failures that are most likely in actual peration, thus directing efforts at fixing the most important faults.

Reliability testing may be performed at several levels. Complex systems may be tested at component, circuit board, unit, assembly, subsystem and system levels.

Software reliability is a key part in software quality. The study of software reliability can be categorized into three parts:

1. Modeling:-Software reliability modeling has matured to the point that meaningful results can be obtained by applying suitable models to the problem.
2. Measurement
3. Improvement

Conclusion:-

Successfully implemented Dining Philosopher problem using Java and MongoDB

ICOER

Computer Laboratory-III

Experiment No. A5

Name:**RAHUL SAWKAR** Roll No: **BA1238**
Branch: Computer Engineering Class: B.E, Sem: II
Date:

Assignment no- A5

Problem Statement:-
A Mobile App for Calculator having Trigonometry functionality is to be designed and tested. The data storage uses 1.text files, 2. XML Use latest open source software modeling, Designing and testing tool/ Scrum-it. Implement the design using HTML-5/Scala/ Python/Java/C++/Rubi on Rails. Perform Positive and Negative testing.

Learning Objective:
1 Implementation of the problem statement using Object oriented programming.
2 Write test cases for Positive and Negative testing.

Theory: Android is an open source and Linux-based operating system for mobile devices such as smartphones and tablet computers. Android was developed by the Open Handset Alliance, led by Google, and other companies.
Android applications are usually developed in the Java language using the Android Software Development Kit.
he basic building block for user interface is a View object which is created from the View class and occupies a rectangular area on the screen and is responsible for drawing and event handling. View is the base class for widgets, which are used to create interactive UI components like buttons, text fields, etc.
The ViewGroup is a subclass of View and provides invisible container that hold other Views or other ViewGroups and define their layout properties.
At third level we have different layouts which are subclasses of ViewGroup class and a typical layout defines the visual structure for an Android user interface and can be created either at run time using View/ViewGroup objects or you can declare your layout using simple XML file main_layout.xml which is located in the res/layout folder of your project.
A layout may contain any type of widgets such as buttons, labels, textboxes, and so on. Following is a simple example of XML file having LinearLayout:

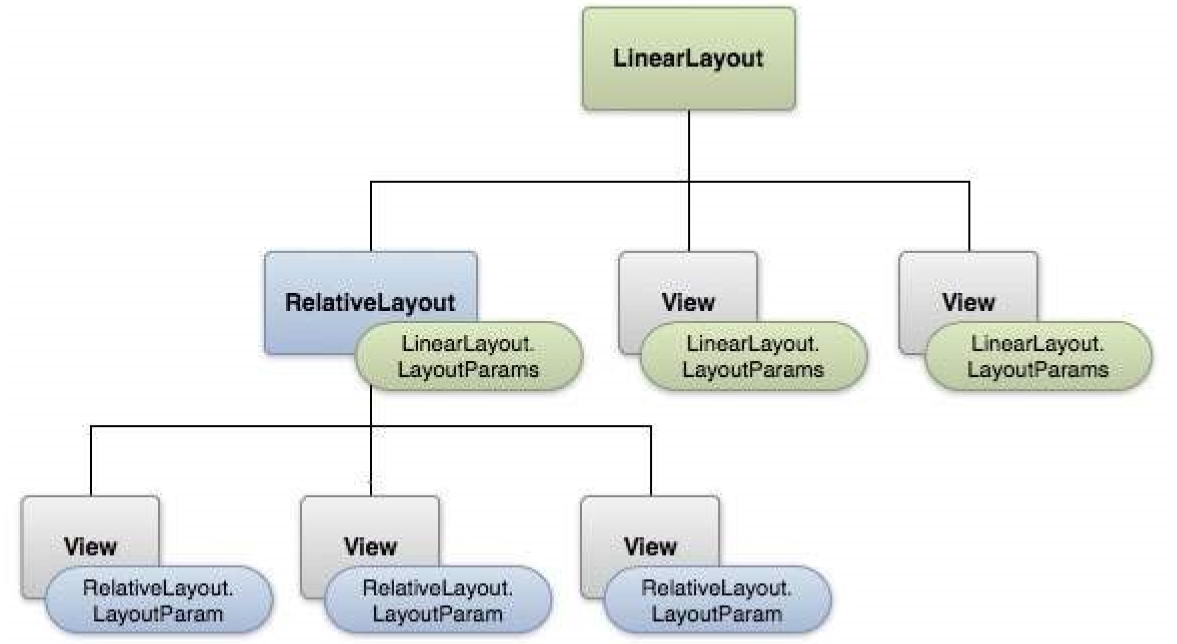
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical" >
```

```
<TextView android:id="@+id/text"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="This is a TextView" />

<Button android:id="@+id/button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="This is a Button" />
<!– More GUI components go here –>
</LinearLayout>
```

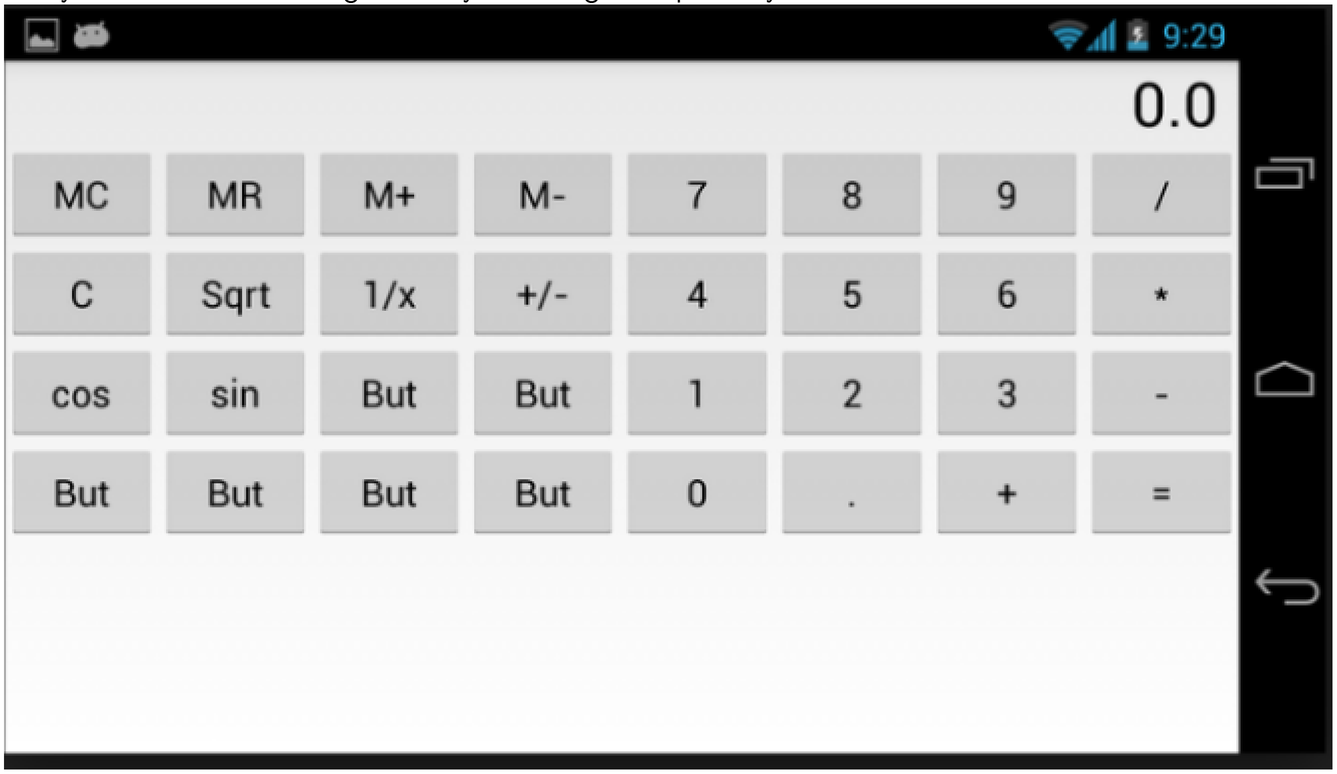
Once your layout has created, you can load the layout resource from your application code, in your Activity.onCreate() callback implementation as shown below

```
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}
```


LinearLayout
RelativeLayout
LinearLayout.LayoutParams
View
LinearLayout.LayoutParams
View
LinearLayout.LayoutParams
View
RelativeLayout.LayoutParam
View
RelativeLayout.LayoutParam
View
RelativeLayout.LayoutParam

Test Scenarios of Calculator
Verify that all the buttons are present and text written on them is readable
Check the arithmetic operations are working fine -,+, -, /, * etc
Verify that BODMAS is applied in case of complex queries and correct result is returned
Verify that calculator gives correct result in case of operations containing decimal numbers
Verify the spacing between the two buttons, the buttons should not be too closely placed
Verify the number of digits allowed to enter in the calculator for any operation
Verify the limit of the response value

Verify the functioning of memory functions
Check if calculator allows navigating through previous calculations
Verify that hitting 'C' cancels any digits or operation added
Verify that on pressing two operators one after the other, the latest one will override the previous operator
Verify the state of calculator when two buttons are pressed simultaneously
Verify if user can delete a digits one by one using backspace key



Test Cases
Check the calculator if it starts by running the application
Check if the calculator application closes if the exit is clicked
Check if all the numbers are working
Check if then sine key is working.
Check if then cosine key is working.
Check if then tan key is working.
Check if then sec key is working.
Check if then cosec key is working.
Check if then cot key is working.
Check the sine of a number.
Check the cosine of a number.
Check the tan of a number.
Check the sec of a number.
Check the cosec of a number.
Check the cot of a number.

Conclusion:-

Successfully Developed A Calculator Application for Android.

Group B

ICOER

Computer Laboratory-III
Experiment No. B1

Name:**RAHUL SAWKAR** Roll No: **BA1238**
Branch: Computer Engineering Class: B.E, Sem: II
Date:

Assignment no- B1

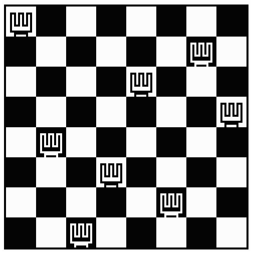
Problem Statement:-
8-Queens Matrix is Stored using JSON/XML having first Queen placed, use back-tracking to place remaining Queens to generate final 8-queen's Matrix using Python..

Learning Objective:
1 Implementation of the problem statement using Object oriented programming.
2 Generate final 8-queen's Matrix using Python.

Theory:
Introduction
The eight queens puzzle is the problem of placing eight chess queens on an 8x8 chessboard so that no two queens threaten each other. Thus, a solution requires that no two queens share the same row, column, or diagonal. The eight queens puzzle is an example of the more general n-queens problem of placing n queens on an nxn chessboard, where solutions exist for all natural numbers n with the exception of n=2 and n=3.
The problem can be quite computationally expensive as there are 4,426,165,368 (i.e., 64C8) possible arrangements of eight queens on an 8x8 board, but only 92 solutions
The eight queens puzzle has 92 distinct solutions. If solutions that differ only by symmetry operations (rotations and reflections) of the board are counted as one, the puzzle has 12 fundamental solution.

- Find an arrangement of 8 queens on a single chess board such that no two queens are attacking one another.
- In chess, queens can move all the way down any row, column or diagonal (so long as no pieces are in the way).

Due to the first two restrictions, it's clear that each row and column of the board will have exactly one queen.



- The backtracking strategy is as follows:
  1. Place a queen on the first available square in row 1.
  2. Move onto the next row, placing a queen on the first available square there (that doesn't conflict with the previously placed queens).
  3. Continue in this fashion until either:
     (a) you have solved the problem, or
     (b) You get stuck.

  When you get stuck, remove the queens that got you there, until you get to a row where there is another valid square to try.
- When we carry out backtracking, an easy way to visualize what is going on is a tree that shows all the different possibilities that have been tried.

Approach:

- Create a solution matrix of the same structure as chess board.
- Whenever place a queen in the chess board, mark that particular cell in solution matrix.
- At the end print the solution matrix, the marked cells will show the positions of the queens in the chess board.

Algorithm:

1. Place the queens column wise, start from the left most column
2. If all queens are placed.
   (a) return true and print the solution matrix.
3. Else
   (a) Try all the rows in the current column.
   (b) Check if queen can be placed here safely if yes mark the current cell in solution matrix as 1 and try to solve the rest of the problem recursively.
   (c) If placing the queen in above step leads to the solution return true.
   (d) If placing the queen in above step does not lead to the solution , BACKTRACK, mark the current cell in solution matrix as 0 and return false.
4. If all the rows are tried and nothing worked, return false and print NO SOLUTION.

Better Solution:

If you notice in solution matrix, at every row we have only one entry as 1 and rest of the entries are 0. Solution matrix takes O(N2) space. We can reduce it to O(N). We will solve it by taking one dimensional array and consider solution[1] = 2 as "Queen at 1st row is placed at 2nd column."

Conclusion:-

Thus we have successfully implemented 8-Queens problem.

ICOER

Computer Laboratory-III

Experiment No. B2

Name:**RAHUL SAWKAR** Roll No: **BA1238**
Branch: Computer Engineering Class: B.E, Sem: II
Date:

Assignment no- B2

Problem Statement:-
A Web application for Concurrent implementation of ODD-EVEN SORT is to be designed using Real time Object Oriented Modeling (ROOM). Give the necessary design diagrams and write the test cases for the white box testing. Draw Concurrent collaboration Diagrams.

Learning Objective:-
1 Understand and Implement the Meaning of Odd-Even sort.
2 Write test cases for White Box Testing.

Theory:-
Introduction:-
In Odd Even Sort compare the Element available on Even Index sort and for Odd sort compare the Element available on Odd index. in this assignment JSP used for Web application development purpose the remaining logic for sorting purpose is bubble sort. JSP technology is used to create web application just like Servlet technology. It can be thought of as an extension to servlet because it provides more functionality than servlet such as expression language, jstl etc. A JSP page consists of HTML tags and JSP tags. The jsp pages are easier to maintain than servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tag etc

Advantage of JSP over Servlet :-
There are many advantages of JSP over servlet. They are as follows:

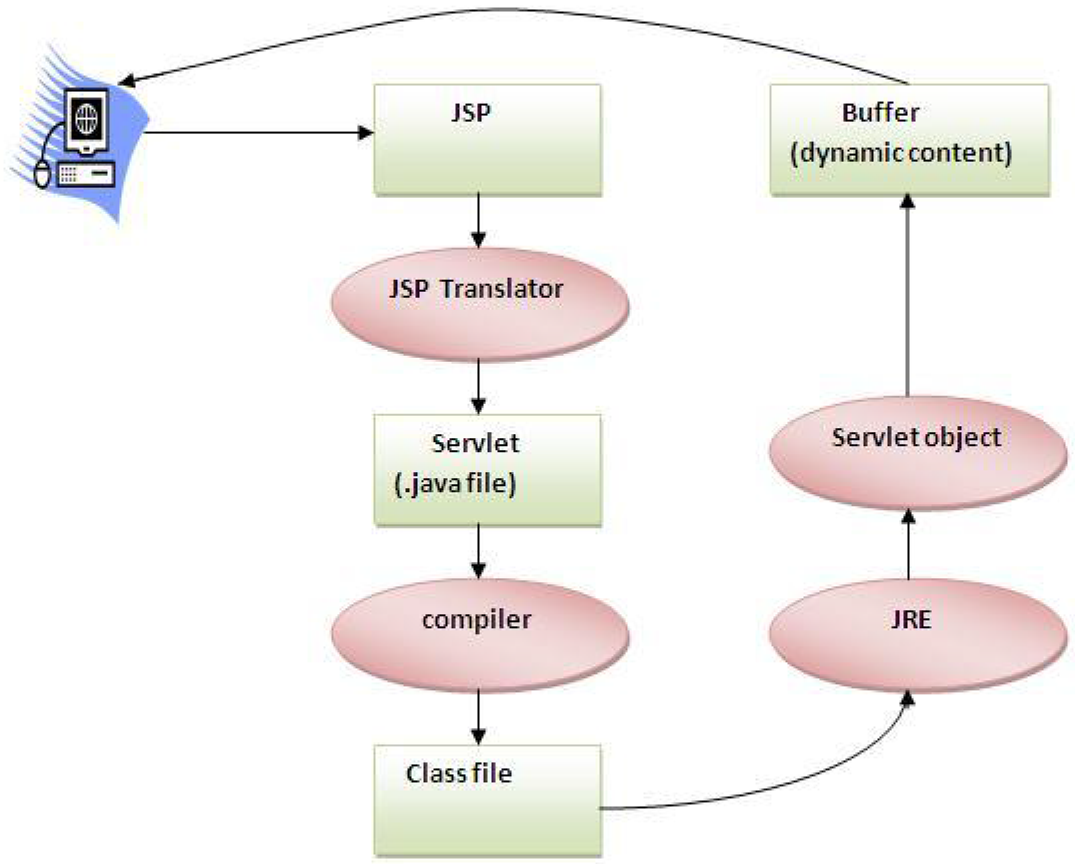
1) **Extension to Servlet**
JSP technology is the extension to servlet technology. We can use all the features of servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.

2) **Easy to maintain**
JSP can be easily managed because we can easily separate our business logic with presentation logic. In servlet technology, we mix our business logic with the presentation logic.

3) **Fast Development: No need to recompile and redeploy**
If JSP page is modified, we don't need to recompile and redeploy the project. The servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

4) **Less code than Servlet**
In JSP, we can use a lot of tags such as action tags, jstl, custom tags etc. that reduces the code. Moreover, we can use EL, implicit objects etc

Life cycle of a JSP Page The JSP pages follows these phases:
Translation of JSP Page
Compilation of JSP Page
Classloading (class file is loaded by the classloader)
Instantiation (Object of the Generated Servlet is created).
Initialization (jspInit() method is invoked by the container).
Reqeust processing (_jspService() method is invoked by the container).
Destroy (jspDestroy() method is invoked by the container).



As depicted in the above diagram, JSP page is translated into servlet by the help of JSP translator. The JSP translator is a part of webserver that is responsible to translate the JSP page into servlet. Afterthat Servlet page is compiled by the compiler and gets converted into the class file. Moreover, all the processes that happens in servlet is performed on JSP later like initialization, committing response to the browser and destroy.
Creating a simple JSP Page:- To create the first jsp page, write some html code as given below, and save it by .jsp extension. We have save this file as index.jsp. Put it in a folder and paste the folder in

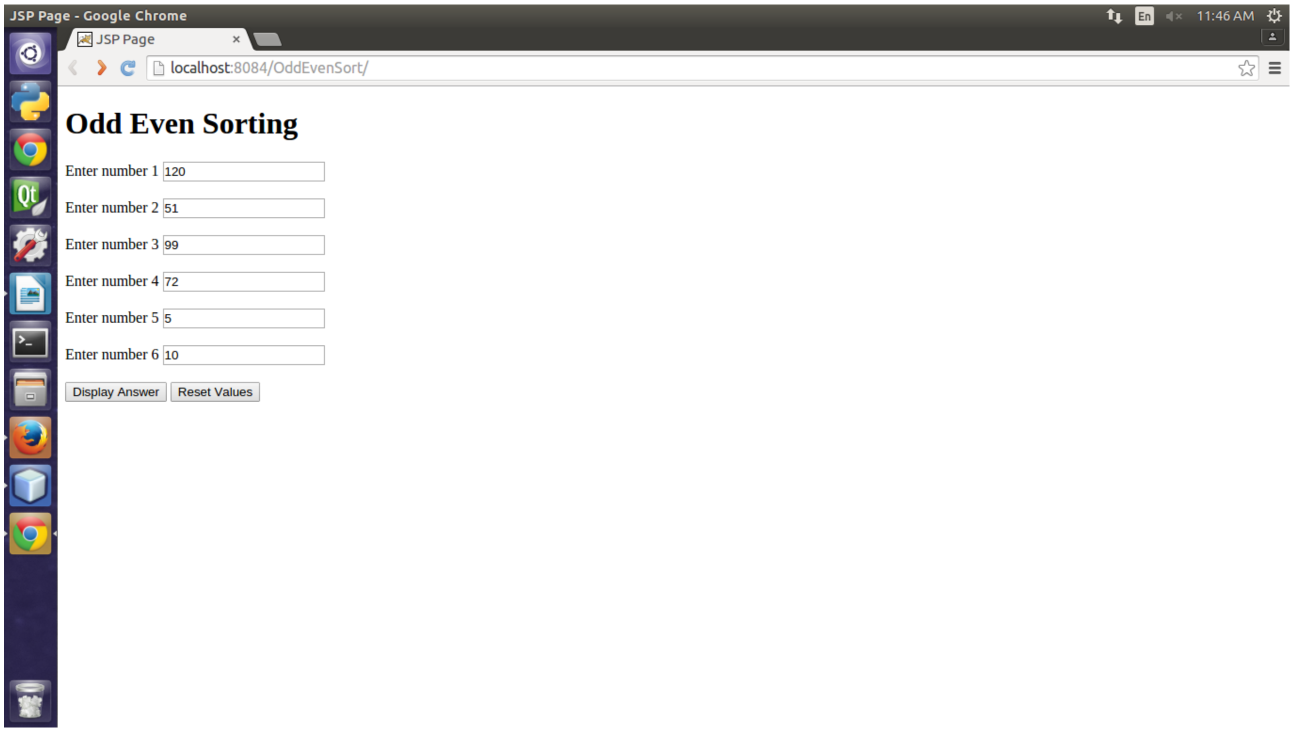
the web-apps directory in apache tomcat to run the jsp page.

index.jsp

```
<%@page contentType="text/html"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body>
<h1>Odd Even Sorting</h1>
<form method="Post" action="OddEven">
Enter number 1
<input type="text" name="no0" value=""/><br><br>
Enter number 2
<input type="text" name="no1" value="" /><br><br>
Enter number 3
<input type="text" name="no2" value="" /><br><br>
Enter number 4
<input type="text" name="no3" value="" /><br><br>
Enter number 5
<input type="text" name="no4" value="" /><br><br>
Enter number 6
<input type="text" name="no5" value="" /><br><br>
<input type="submit" value="Display Answer"/>
<input type="reset" value="Reset Values"/>
</form>
</body>
</html>
```

How to run a simple JSP Page ?

Follow the following steps to execute this JSP page:
Start the servers put the jsp file in a folder and deploy on the server visit the browser by the url http://localhost:portno/contextRoot/jspfile



Odd Even Sorting

Enter number 1 120

Enter number 2 51

Enter number 3 99

Enter number 4 72

Enter number 5 5

Enter number 6 10

e.g. http://localhost:8084/OddEven/

Conclusion:-

Successfully developed A Web application for Concurrent implementation of ODD-EVEN SORT

ICOER

Computer Laboratory-III

Experiment No. B4

Name: **RAHUL SAWKAR** Roll No: **BA1238**
Branch: Computer Engineering Class: B.E, Sem: II
Date:

Assignment no- B4

Problem Statement:
Write a web application using Java to check the plagiarism in the given text paragraph written/ copied in the text box. Give software Modeling, Design, UML and Test cases for the same using COMET (Concurrent Object Oriented Modeling and Architectural Design Method).

Learning Objective:
1 Understand the Meaning of Software modeling using COMET.
2 Implement the logic for Check the Plagiarism in the given text.

Theory:

Java Scanner class:
To Check the Plagiarism in the given text first of all perform string compare operation also need to understand the operation related to scanner classes.There are various ways to read input from the keyboard, the java.util.Scanner class is one of them.

- The Java Scanner class breaks the input into tokens using a delimiter that is whitespace bydefault.It provides many methods to read and parse various primitive values.
- Java Scanner class is widely used to parse text for string and primitive types using regular expression.
- Java Scanner class extends Object class and implements Iterator and Closeable interfaces.

There is a list of commonly used Scanner class methods:

Method	Description
public String next()	it returns the next token from the scanner.
public String nextLine()	it moves the scanner position to the next line and returns the value as a string.
public byte nextByte()	it scans the next token as a byte.
public short nextShort()	it scans the next token as a short value.
public int nextInt()	it scans the next token as an int value.
public long nextLong()	it scans the next token as a long value.
public float nextFloat()	it scans the next token as a float value.
public double nextDouble()	it scans the next token as a double value.

java.io.PrintStream class

The PrintStream class provides methods to write data to another stream. The PrintStream class automatically flushes the data so there is no need to call flush() method. Moreover, its methods don't throw IOException.

Stream

A stream is a sequence of data.In Java a stream is composed of bytes. It's called a stream because it's like a stream of water that continues to flow. In java, 3 streams are created for us automatically. All these streams are attached with console.
1) System.out: standard output stream
2) System.in: standard input stream
3) System.err: standard error stream

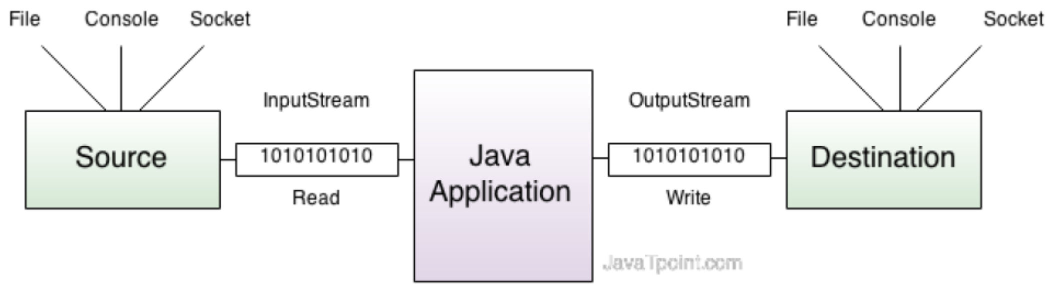
Let's see the code to print output and error message to the console.
1. System.out.println("simple message");
2. System.err.println("error message");

OutputStream

Java application uses an output stream to write data to a destination, it may be a file,an array,peripheral device or socket.

InputStream

Java application uses an input stream to read data from a source, it may be a file,an array,peripheral device or socket.



Output Stream class

OutputStream class is an abstract class.It is the superclass of all classes representing an output stream of bytes. An output stream accepts output bytes and sends them to some sink.

Reading data from keyboard

There are many ways to read data from the keyboard.
For example:

- InputStreamReader
- Console Scanner
- DataInputStream etc.

InputStreamReader class

InputStreamReader class can be used to read data from keyboard.It performs two tasks:

- connects to input stream of keyboard
- converts the byte-oriented stream into character-oriented stream

Java Console class

The Java Console class is be used to get input from console. It provides methods to read text and password. If you read password using Console class, it will not be displayed to the user.
The java.io.Console class is attached with system console internally.The Console class is introduced since 1.5.
Let's see a simple example to read text from console.
1. String text=System.console().readLine();
2. System.out.println("Text is: "+text);

FileInputStream and FileOutputStream (File Handling)

In Java, FileInputStream and FileOutputStream classes are used to read and write data in file. In another words, they are used for file handling in java.

Java FileOutputStream class

Java FileOutputStream is an output stream for writing data to a file.

Java String compare To

The java string compareTo() method compares the given string with current string lexicographically. It returns positive number, negative number or 0. If first string is greater than second string, it returns positive number (difference of character value). If first string is less than second string, it returns negative number and if first string is equal to second string, it returns 0.
1. s1 > s2 => positive number
2. s1 < s2 => negative number

3. s1 == s2 => 0

COMET

COMET is a highly iterative object-oriented software development method that addresses the requirements, analysis, and design modeling phases of the object-oriented development life cycle. The functional requirements of the system are defined in terms of actors and use cases. Each use case defines a sequence of interactions between one or more actors and the system. A use case can be viewed at various levels of detail. In a requirements model, the functional requirements of the system are defined in terms of actors and use cases. In an analysis model, the use case is realized to describe the objects that participate in the use case, and their interactions.

Conclusion:

Thus, we have successfully designed a web application using Java to check the plagiarism in the given text paragraph written/ copied in the text box.

Group C

ICOER

Computer Laboratory-III
Experiment No. C2

Name:**RAHUL SAWKAR** Roll No: BA1238
Branch: Computer Engineering Class: B.E, Sem: II
Date:

Assignment no- C2

Problem Statement:
Install and Use Latest IDS (Open Source).

Introduction to IDS:

An intrusion detection system (IDS) is a device or software application that monitors network or system activities for malicious activities or policy violations and produces reports to a management station.
IDS come in a variety of "flavors" and approach the goal of detecting suspicious traffic in different ways. There are network based (NIDS) and host based (HIDS) intrusion detection systems.

A. NIDS:

It is a network security system focusing on the attacks that come from the inside of the network (authorized users). It detects attacks as they happen. It is monitoring Real-time networks and also provides information about attacks that have succeeded. It deploy sensors at strategic locations E.G., Packet sniffing via tcpdump at routers.

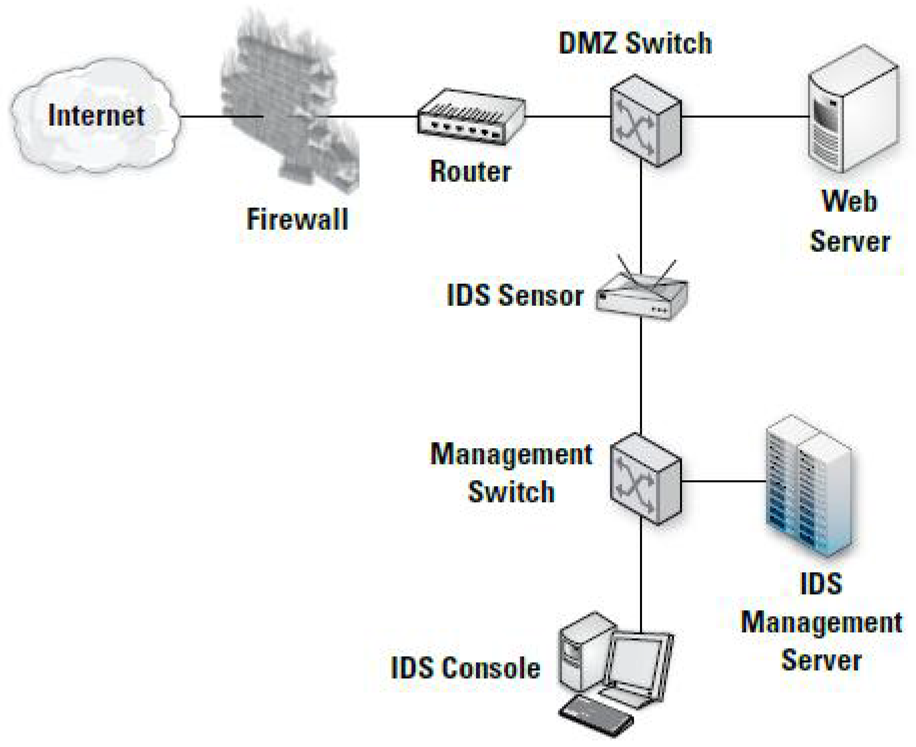


Figure 2 NIDS placement

Advantages of NIDS:

1) Easier to deploy: Network based IDS are easier to deploy as it does not affect existing systems or infrastructure.
2) Detect network based attacks: Network based IDS sensors can detect attacks, which host based sensors fail to detect. A network based IDS checks for all the packet headers for any malicious attack.
3) Detection of failed attacks: A network based IDS sensor deployed outside the firewall can detect malicious attacks on resources behind the firewall, even though the firewall may be rejecting these attempts.

B. HIDS:

It is an intrusion detection system that monitors and analyzes the internals of a computing system as well as (in some cases) the network packets on its network interfaces (just like a network-based intrusion detection system (NIDS) would do).It monitors user activities E.G., Analyze shell commands

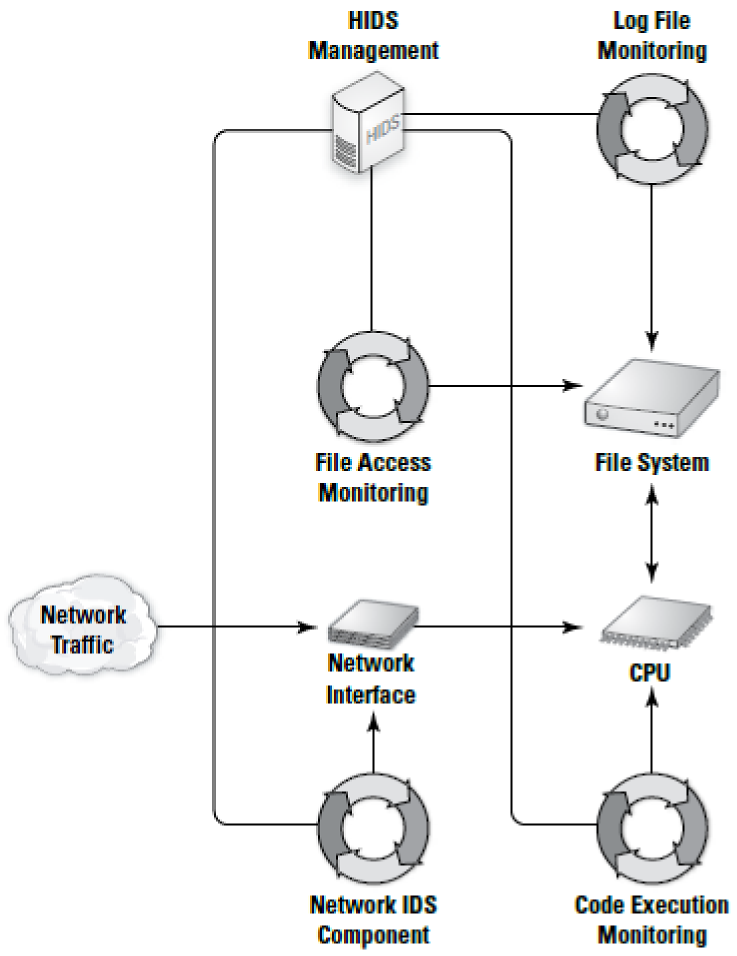


Figure 4 HIDS block diagram

Advantages of HIDS:

1) Verifies success or failure of an attack: Since a host based IDS uses system logs containing events that have actually occurred, they can determine whether an attack occurred or not with greater accuracy and fewer false positives than a network based system.
2) Detects attacks that a network based IDS fail to detect: Host based systems can detect attacks that network based IDS sensors fail to detect.
3) Lower entry cost: Host based IDS sensors are far more cheaper than the network based IDS sensors.

Functions of IDS:

Monitoring users and system activity

- auditing system configuration for vulnerabilities and misconfigurations
- assessing the integrity of critical system and data files
- recognizing known attack patterns in system activity

TOOLS:

1) Snort Alert Monitor:

- Java based console.
- Will give a quick look at the Snort alert
- Can be configured to send e-mail when Snort alerts to an attempted exploit on your network.

2) Snortalog:

- It is a Perl based Snort log analyser.
- Allows to develop plaint text on HTML summery reports and graph representation of top attack that has been detected by Snort Sensor

3) SnortFW:

- It analyses incoming Snort alerts and updates iptables firewall to block the attacker.

4) IDSCenter:

It is an all-in-one centralized graphical utility for managing.

- Snort
- Alerts
- Rules
- Configuration files
- Distributing updates
- Generate reports
- Email
- Auditable/visual alarm notification.

5) Wireshark:

Wireshark is a free and open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development.

IDS using Wireshark:

Conclusion:

Thus, we have successfully installed and Used Latest IDS (Open Source)