Experiment no. 01

Name of the Experiment: Write a Java or C program to develop a simple calculator that would be able to take a number, an operator (addition/subtraction/multiplication/division/modulo) and another number consecutively as input and the program will display the output after pressing “=” sign.

Algorithm

```
1. BEGIN
2. ENTER two integers separated by an operator and suffixed with a semicolon.
3. IF the suffix is not a semicolon but “=” go to step ……….
   ELSE continue
4. ENTER integers in a vector nums
5. ENTER the operator in another vector ops
6. IF op = “+”, ENTER the summation of the integers in result vector
   ELSE IF op = “-”, ENTER the subtraction of the integers in result vector
   ELSE IF op = “*”, ENTER the multiplication of the integers in result vector
   ELSE IF op = “/”,
           IF the second operand is zero print “Invalid”
           ELSE ENTER the result of division of the integers in result vector
   ELSE IF op = “%”, ENTER the modulo of the integers in result vector
7. PRINT
```

Source Code

```
#include <bits/stdc++.h>
using namespace std;
#define lli long long int

int main(){
    lli a,b;
    char op,ch;
    vector<lli> nums;
    vector<char> ops;
    vector<lli> result;
    while(1){
        cin>>a>>op>>b>>ch;
        nums.push_back(a);
        nums.push_back(b);
        ops.push_back(op);
        if(op=='+') result.push_back(a+b);
        else if(op=='-') result.push_back(a-b);
        else if(op=='*') result.push_back(a*b);
        else if(op=='/') result.push_back(a/b);
        else if(op=='%') result.push_back(a%b);

        if(ch=='=') break;
    }
    lli len = ops.size();
    for(lli i=0,j=0;i<len;i++){
                  cout<<nums[j]<<" "<<ops[i]<<" "<<nums[j+1]<<" =
"<<result[i]<<endl;
        j+=2;
    }
}
```

INPUT	OUTPUT

1+2;
4*3;
15/5;
21%4;
76-56
=
	1 + 2 = 3
	4 * 3 = 12
	15 / 5 = 3
	21 % 4 = 1
	76 - 56 = 20

Experiment no. 02

Name of the Experiment: Write a Java or C program that will take two ‘n’ integers as input until a particular operator and produce ‘n’ outputs.

Algorithm

```
1. BEGIN
2. WHILE the input stream consists of integers
       ENTER integers
3. IF number of integers is ODD, ABORT
4. SWITCH (character)
    a. Case +
        i.  Print summation of two numbers
        ii. Iterate next two numbers
    b. Case -
        i.  Print subtraction of two numbers
        ii. Iterate next two numbers
    c. Case *
        i.  Print multiplication of two numbers
        ii. Iterate next two numbers
    d. Case /
       IF the second operand is zero, PRINT invalid
       ELSE
        i.  Print division of two numbers
        ii. Iterate next two numbers
    e. Case %
        i.  Print modulo of two numbers
        ii. Iterate next two numbers
5. END
```

SOURCE CODE

```
#include <bits/stdc++.h>
using namespace std;
#define lli long long int

int main(){
    char ch;    lli x; string s;
    vector<lli> nums;
    while(1){
        cin>>s;
        if(s=="+" or s=="-" or s=="*" or s=="/" or s=="%") break;
        istringstream iss(s);
        iss>>x; nums.push_back(x);
    }
    lli len = nums.size();
    if(len%2) cout<<"Invalid length of numbers\n";
    else{
        if(s=="+"){
            for(lli i=0;i<len;i+=2) cout<<nums[i]+nums[i+1]<<" ";}
        if(s=="-"){
            for(lli i=0;i<len;i+=2) cout<<nums[i]-nums[i+1]<<" ";}
        if(s=="*"){
            for(lli i=0;i<len;i+=2) cout<<nums[i]*nums[i+1]<<" ";}
        if(s=="/"){
            for(lli i=0;i<len;i+=2) {
                if(nums[i+1]==0) cout<<"invalid ";
                else cout<<nums[i]/nums[i+1]<<" ";}}
        if(s=="%"){
            for(lli i=0;i<len;i+=2) cout<<nums[i]%nums[i+1]<<" ";}
    }}
```

INPUT	**OUTPUT**
1 2 3 4 5 6 7 8 + 7 5 4 1 12 19 32 12 - 1 4 2 *	3 7 11 15 2 3 -7 20 invalid

Experiment no. 03

Name of the Experiment: Write a program in Java or C to check whether a number or string is palindrome or not.

ALGORITHM

1. Find length of str. Let length be n.
2. Initialize low and high indexes as 0 and n-1 respectively.
3. Do the following while the low index ‘l’ is smaller than the high index ‘h’.
   i. If str[l] is not the same as str[h], then return false.
   ii. Increment l and decrement h, i.e., do l++ and h–.
4. If we reach here, it means we didn’t find a mis

SOURCE CODE

```
#include <bits/stdc++.h>
using namespace std;
#define lli long long int

string isPalindrome(string S)
{
    for (int i = 0; i < S.length() / 2; i++) {
        if (S[i] != S[S.length() - i - 1]) {
            return "No. This is not a palindrome";
        }
    }
    return "Yes, this is a palindrome";
}

int main(){
    string s;
    cin>>s;
    cout<<isPalindrome(s);
}
```

INPUT	**OUTPUT**
thesis malaylam	No. This is not a palindrome Yes, this is a palindrome

PROBLEM NO. : 04

NAME OF THE PROBLEM: Write down the ATM system specifications and report the various bugs.

Description: An automated teller machine (ATM) is a computerized telecommunications device that provides the customers of a financial institution with access to financial transactions in a public space without the need for a human clerk or bank teller. On most modern ATMs, the customer is identified by inserting a plastic ATM card with a magnetic stripe or a plastic smartcard with a chip, that contains a unique card number and some security information, such as an expiration date or CVC (CVV). Security is provided by the customer entering a personal identification number (PIN).

Using an ATM, customers can access their bank accounts in order to make cash withdrawals (or credit card cash advances) and check their account balances. The functions of the system are:

1. Login
2. Get Balance Information
3. Withdraw Cash
4. Transfer Funds.

The hardware,software and technology used should have following specifications:

1. Ability to read the ATM card.
2. Ability to count currency notes.
3. Touch screen for convenience.
4. Keypad(in case touchpad fails)
5. Continuous power supply
6. Ability to connect to the bank network.
7. Ability to validate users.

Hardware interfaces :
The hardware should have following specifications:

1. Ability to read the ATM card
2. Ability to count the currency notes
3. Touch screen for convenience
4. Keypad (in case touchpad fails)
5. Continuous power supply
6. Ability to connect to bank„s network
7. Ability to take input from user
8. Ability to validate user

Some of the possible Bugs on the ATM machine:

1. Successful insertion of ATM card
2. Unsuccessful operation due to insert card in wrong angle
3. Unsuccessful operation due to invalid account Ex: other bank card or time expired card
4. successful entry of PIN number 5. un successful operation due to entering the wrong PIN number 3 times
6. successful selection of language
7. successful selection of account type
8. unsuccessful operation due to invalid account type
9. successful selection of withdrawal operation
10. successful selection of amount to be withdrawn
11. successful withdrawal operation.
12. unsuccessful withdrawal operation due to wrong denominations
13. unsuccessful withdraw operation due to amount is greater than day limit
14. unsuccessful withdraw operation due to lack of money in ATM
15. unsuccessful withdrawal operation due to an amount greater than possible balance.

Experiment no. 05

Name of the Experiment: Write a program in Java or C to find out the factorial of a number using while for loop. Also verify the results obtained from each case.

ALGORITHM

1. Create an array *res* of MAX size where MAX is a number of maximum digits in output.
2. Initialize the value stored in res as 1 and initialize res_size (size of ‘res[]’) as 1.
3. Multiply x with *res* and update res and res_size to store the multiplication result for all the numbers from x = 2 to n.
4. To multiply a number x with the number stored in res[], one by one multiply x with every digit of *res*.
5. To implement multiply function perform the following steps:
    a. Initialize carry as 0.
    b. Do following for i = 0 to res_size – 1
        - Find value of res[i] * x + carry. Let this value be prod.
        - Update res[i] by storing the last digit of prod in it.
        - Update carry by storing the remaining digits in carry.

    c. Put all digits of carry in res[] and increase res_size by the number of digits in carry.

CODE

```
#include <bits/stdc++.h>
using namespace std;
#define lli long long int
#define MAX 500
int multiply(int x, int res[], int res_size);
void factorial(int n)
{
      int res[MAX];
      res[0] = 1;
      int res_size = 1;
      for (int x = 2; x <= n; x++) res_size = multiply(x, res,
res_size);
      cout << "Factorial of given number is \n";
      for (int i = res_size - 1; i >= 0; i--) cout << res[i];
}
int multiply(int x, int res[], int res_size)
{
      int carry = 0;
      for (int i = 0; i < res_size; i++) {
            int prod = res[i] * x + carry;
            res[i] = prod % 10;
            carry = prod / 10;
      }
      while (carry) {
            res[res_size] = carry % 10;
            carry = carry / 10;
            res_size++;
      }
      return res_size;
}
int main()
{
    lli x;
    cin>>x;
      factorial(x);
}
```

INPUT	**OUTPUT**
13	Factorial of given number is 6227020800

Experiment no. 06
Name of the Experiment: Write a program in Java or C that will find the sum and average of an array using the do while loop and 2 user defined functions.

ALGORITHM

1. The program takes an array of elements and stores them in an array.
2. Using a for loop, the sum of the array is calculated.
3. The result is printed.
4. Using a for loop, the average of the array is calculated.
5. EXIT

CODE

```
#include <bits/stdc++.h>
using namespace std;
#define lli long long int
#define MAX 500

lli sum(lli arr[], lli n)
{
    lli sum = 0;
    for (lli i = 0; i < n; i++) sum += arr[i];
    return sum;
}
double average(lli a[], lli n)
{
    lli sum = 0;
    for (lli i=0; i<n; i++)      sum += a[i];
    return (double)sum/n;
}
int main()
{
    int n; lli arr[n];
    for(lli i=0;i<n;i++) cin>>arr[i];
    cout << "Sum of given array is " << sum(arr,n)<<endl;
    cout << "Average of given array: "<<average(arr, n) << endl;
    return 0;
}
```

INPUT	**OUTPUT**

6
1 2 3 4 5 6
	`Sum of given array is 21`
	`Average of given array: 3.5`

PROBLEM NO : 07

NAME OF THE PROBLEM: Write a simple “JAVA” program to explain classNotFound Exception and endofFile(EOF) exception.

Description: This exception indicates the end of file (EOF), or the end of stream has been reached unexpectedly. Also, this exception is mainly used by DataInputStreams, in order to signal the end of the stream. However, notice that other input operations may return a special value upon the end of a stream, instead of throwing an EOFException. The EOFException class extends the IOException class, which is the general class of exceptions produced by failed, or interrupted I/O operations. Moreover, it implements the Serializable interface. Also, it is defined as a checked exception and thus, it must be declared in a method, or a constructor’s throws clause.

DataInputStreams provide methods that can read **primitive** Java data types from an underlying input stream in a machine-independent way. An application writes data, by using the methods provided by the OutputStream class, or the DataOutputStream class.
Specifically, primitive types can be read by an application, using one of the following methods:

- **readBoolean()** – Reads one input byte and returns true if that byte is nonzero, false if that byte is zero.
- **readByte()** – Reads and returns one input byte.
- **readChar()** – Reads two input bytes and returns a char value.
- **readDouble()** – Reads eight input bytes and returns a double value.
- **readFloat()** – Reads four input bytes and returns a float value.
- **readInt()** – Reads four input bytes and returns an int value.
- **readLong()** – Reads eight input bytes and returns a long value.
- **readShort()** – Reads two input bytes and returns a short value.
- **readUnsignedByte()** – Reads one input byte and returns it as a zero-extended int value. The integer value resides in the range [0, 255].
- **readUnsignedShort()** – Reads two input bytes and returns them as an int value. The integer value resides in the range [0, 65535].

<u>Source Code:</u>

```
import java.io.*
 public class EOFExceptionExample {
    private final static String FILENAME = "input.txt";
    private static void writeToFile() throws IOException {
DataOutputStream out = new DataOutputStream(new FileOutputStream(FILENAME));
        String str = "Hello from john dalton";
        for(int i = 0; i < str.length(); ++i)
            out.writeChar(str.charAt(i));
        out.close();return;
    }
    public static void main(String[] args) {
        DataInputStream input = null;
        try {
            writeToFile();
            input = new DataInputStream(new FileInputStream(FILENAME));
            while(true) {
                char num;
                try {
                    num = input.readChar();
                    System.out.println("Reading from file: " + num);}
                catch (EOFException ex1) {
                    break; //EOF reached.}
                catch (IOException ex2) {
                    System.err.println("An IOException was caught: " + ex2.getMessage());
                    ex2.printStackTrace();}}}
        catch (IOException ex) {
            System.err.println("An IOException was caught: " + ex.getMessage());
            ex.printStackTrace();
        }
        finally {
            try {
                // Close the input stream.
                input.close();
            }
            catch(IOException ex) {
                System.err.println("An IOException was caught: " + ex.getMessage());
                ex.printStackTrace();
            }
        }
    }}
```

Output:

```
Reading from file: H
Reading from file: e
Reading from file: l
Reading from file: l
Reading from file: o
Reading from file:
Reading from file: f
Reading from file:
Reading from file: f
Reading from file: r
Reading from file: o
Reading from file: m
Reading from file:
Reading from file: j
Reading from file: h
Reading from file: o
Reading from file: n
Reading from file:
Reading from file: d
Reading from file: a
Reading from file: l
Reading from file: t
Reading from file: o
Reading from file: n
```

PROBLEM NO: 08

NAME OF THE EXPERIMENT: To write a C or Java program that will read input.txt file containing n positive integer and calculate addition,subtraction, multiplication and division in separate output.txt files.

ALGORITHM

1. START
2. Open input.txt file and assign in f;
3. Now, read input text file and assign it into another variable
4. Print read file
5. Open output.txt
6. Use if condition and for loop for performing calculation
7. Display output and close file by using close()
8. Save and Exit

SOURCE CODE

```
#include<bits/stdc++.h>
typedef long long int ll;
using namespace std;

int main()
{
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
    double x;
    vector<double>nums;
    while(x!=-1){
        cin>>x; nums.push_back(x);
    for(int i=0; i<nums.size()-1; i+=2)
    {
        double num1 = nums[i], num2 = nums[i+1];
        cout<<num1<<" + "<<num2<< " = "<<num1+num2<<endl;
        cout<<num1<<" - "<<num2<< " = "<<num1-num2<<endl;
        cout<<num1<<" * "<<num2<< " = "<<num1*num2<<endl;
        if(num2==0){
  cout<<"Invalid input for division and modulation operation."<<endl;
        }
        else cout<<num1<<" / "<<num2<< " = "<<num1/num2<<endl;
        if(num2==0) cout<<"Invalid input."<<endl;
     else cout<<num1<<" % "<<num2<< " = "<<(int)num1%(int)num2<<endl;
        }
        cout<<endl;
    }
    return 0;
}
```

INPUT	OUTPUT
5 5 8 9	Case 1: 10 0 25 1 Case 2: 17 1 72 1

PROBLEM NO. : 09

NAME OF THE PROBLEM: Explain the role of Software Engineering in Biomedical Engineering and in the field of Artificial Intelligence and Robotics.

Description: Artificial intelligence has wide applications in the field of biomedical engineering. In Fact it will transform healthcare in the near future. Following are some of its applications:

Health Apps: There are various apps on your phone which use AI like google assistant but there are also Some health apps like **CareBox** Your health companion app which use AI. **CareBox** learns as you go, asking smart questions to help you feel better and take control of your health.It predicts if you are having disease based on your symptoms.

Expert systems : In artificial intelligence, an expert system is a computer system that emulates the decision-making ability of a human expert. Expert systems like MYCIN for **bacterial diseases** and CaDET for **cancer detection** are widely used.

Image processing: Image processing is very critical when it comes to healthcare because we have to detect disease based on the images from x-ray MRI and CT scans so an AI system that detects those minute tumor cells which is really handy in early detection of diseases.

Surgical Robots: The most interesting and definitely a revolutionary invention as it can change surgery completely with robots that learn from their mistakes. We can have surgical Robots with high precision, performing surgeries in your local hospital and that's what I am looking forward to.

Artificial intelligence is a term for computer systems that can perform tasks that require human intelligence and discernment, like the ability to reason, perceive, and generalize. The computer should be able to sense its environment and take action according to what it learns.

AI algorithms can improve project planning, aid with automation QA (quality assurance), and enhance user experience. A recent report found that AI-enhanced software development increased the productivity of a developer by 10 times.

Here are some ways AI can power your software development and deployment processes by automating various cognitive and physical tasks.

1. Increase in the Speed and Scale of Development
2. Changing The Role of Developers
3. Strategic Decision-Making
4. Error Management
5. Precise Estimates
6. Connect to Real-Time Feedback

PROBLEM NO. : 10

NAME OF THE PROBLEM: Explain the various phases of water-fall model, which phase is the most dominated one?

Description: A waterfall process model is a dominant software development model used by software engineers for the last three decades. This model divides the entire software development process into several independent stages. Further, the developers sequentially carry out these independent stages to obtain the final product (software).

- For the last three decades, the waterfall model has been a popular software development model.
- The waterfall model divides the entire process of software development into definite independent stages.
- Each stage is well defined and produces a precise output passed on to the next stage as an input.
- The stages of the waterfall model are requirement analysis, planning, modeling, coding, testing & deployment.
- In the requirement analysis, the developer expects the client to specify all the requirements explicitly.
- In the planning stage, the planning is done by scheduling the remaining stages, estimating the cost of each stage, and determining the hardware to be used.
- At the modeling stage, developers design logical models.
- In the coding stage, the developer implements a logical model by developing small integrated programs to form the entire software.
- The software developed is tested in a customer-friendly environment to verify that no flaws remain in the software in the testing stage.
- The product or the software is handed over to the client in the deployment stage.

Maintenance: It is the last but the most important phase in the waterfall workflow model. This step comes just after installation, and it includes making the appropriate modification to the product or system or enhancing, changing, or modifying attributes related to performance issues related to the system.

PROBLEM NO. : 11

NAME OF THE PROBLEM: Using COCOMO Model estimate effort for Specific problem in Industrial Domain.

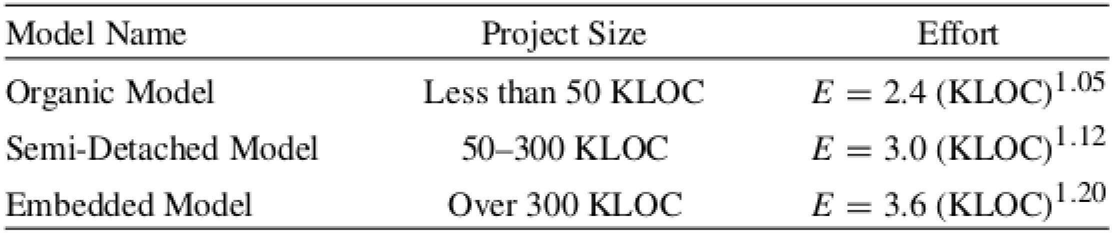
Description: Optimizing Basic COCOMO Model using Simplified Genetic Algorithm

The Constructive Cost Model (COCOMO):

The Constructive Cost Model is a well documented and widely accepted algorithmic model for effort estimation. It was developed by Barry W. Boehm in 1981 . The Constructive Cost Model has three variants. These are basic, intermediate and detailed. The main parameter for the SEE is the size of the project. The size is represented in terms of lines of code (LOC) or a thousand-lines of code (KLOC). This model was built based on a historical dataset of 63 projects. The model defines mathematical equations for estimating effort, development time and the maintenance effort. The mathematical equation of a basic COCOMO model is defined in Equation

$$E = A \times (KLOC)\hat{}\ B \qquad (1)$$

Table 1. Basic-COCOMO Models.



Model Name	Project Size	Effort
Organic Model	Less than 50 KLOC	$E = 2.4\ (\text{KLOC})^{1.05}$
Semi-Detached Model	50–300 KLOC	$E = 3.0\ (\text{KLOC})^{1.12}$
Embedded Model	Over 300 KLOC	$E = 3.6\ (\text{KLOC})^{1.20}$

where KLOC is the size of the code (kilo-lines of code), E is the software effort computed in person-month and A, B are the COCOMO model parameters.

The value of A and B depend on the model of a software project. COCOMO model has three types, depending upon project size. These models are Organic, Semi-detached and Embedded models. Table 1 shows the basic COCOMO models. The main issue with the COCOMO model is that it does not provide realistic effort in the current development environment. This limitation of the COCOMO model was overcome by exploration of non-algorithmic techniques like genetic algorithms or any other nature-inspired algorithms. Software effort estimation based on existing parameters does not always give precise results; due to that, we require the tuning of the parameters to get more accurate results.

Genetic Algorithm (GA) : The genetic algorithm is an evolutionary search algorithm. It belongs to the field of nature-inspired algorithms. In the 1970s, John Holland and his collaborators proposed a genetic algorithm at University of Michigan. It is a global optimization method which is inspired from the abstraction of Darwinian evolution and natural selection of biological systems. GA uses mathematical operators such as selection of fittest, crossover and mutation. The wide range of optimization problems have been successfully solved using genetic algorithms.

The parameters of the basic COCOMO model are optimized using the simplified genetic algorithm. This shall ensure that complexity of the proposed algorithm remains low. We use the crossover and selection operator for calculating new values of parameters A and B.

The steps of the proposed simplified genetic algorithm are:

1. Randomly generate an initial population of A and B.
2. Calculate the fitness of each chromosome using the fitness function.
3. Select the parent chromosome from the population.
4. Create new chromosomes by applying crossover for A after 8th bit to 17th bit and for B after 7th bit till 15th bit.
5. Compute the fitness of new chromosomes.
6. Select the next population by selecting the best chromosomes.
7. Goto step 3, until max-generation is not reached else stop and return the best chromosome.

We propose to use the absolute difference between effort and estimated effort (Manhattan distance (MD)) as the fitness function for the proposed simplified genetic algorithm.

1. Manhattan Distance,
2. Mean Magnitude of Relative Error (MMRE)
3. Variance-Accounted-For (VAF)
4. Root Mean Square (RMS)
5. Euclidean Distance (ED)

The proposed experiment applies simplified GA for optimizing the value of A and B of the basic COCOMO model given in Equation 1. This model allows us to estimate effort for the software development of the 18 projects of NASA dataset. The parameters of simplified GA are set as Table 3. The fitness function is shown in Equation 2. The computed parameters can significantly simplify the estimation of the software effort for all projects.

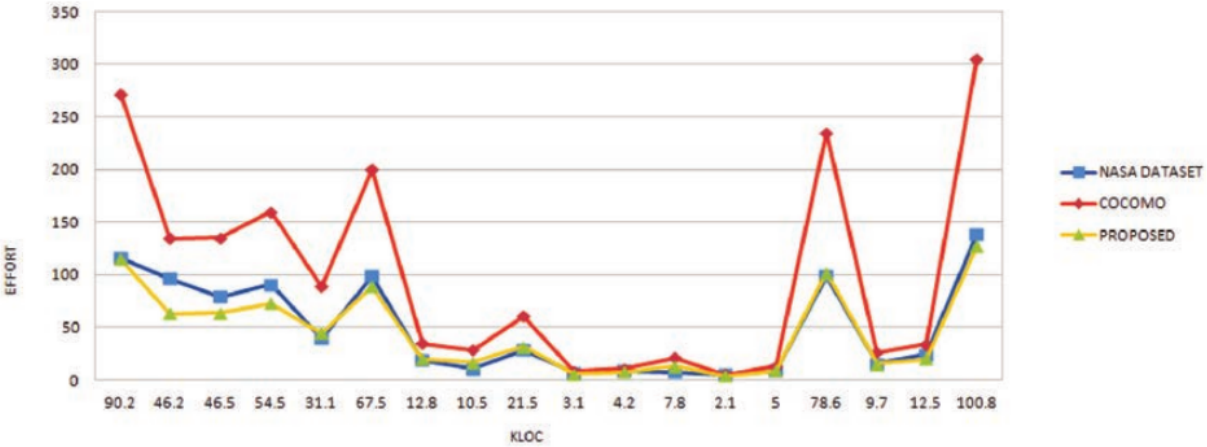


Fig. 1. Effort Graph for NASA, COCOMO and Proposed Simplified GA.

better results as compared to actual NASA effort. The performance comparison of basic COCOMO and proposed simplified GA

$$E = 2.022817 \times (KLOC)^{0.897183}$$

This paper proposes a software effort estimation (SEE) model using a simplified form of genetic algorithm. The simplified GA is used for optimizing the parameters of the basic COCOMO model. This is computationally cheaper also. The performance analysis of the proposed model is analyzed on the well-known NASA dataset. We have used the simplified genetic algorithm for developing a model for the software effort calculation. The Manhattan distance of proposed simplified GA is 6.7125 which is better than basic COCOMO.

PROBLEM NO: 12

NAME OF THE PROBLEM: To identify the reason behind software crises and explain the possible solution for following scenario

Case 01: Air ticket reservation software was delivered to the customer and was installed in an airport at 12 AM as per plan. The system worked quite fine till the next day at 12:00 PM. The system crashed at 12:00 PM and the airport authorities could not continue using software for reservation till 5:00 PM. It took five hours to fix the defect in the software.

Case 02: Software for the financial system was delivered to the customer. Customer confirmed the development team about a function in the system. As the software was huge and complex, the development team could not identify the defect in the software.

Solution: These situations are collectively termed as software crisis

i) Time slippage
ii) Cost slippage
iii) Failure custom site
iv) Intractable error

Case 01 solution: In the crisis of the software online ticket reservation “Failure at customer site”. In this situation the customer cannot do their checking or other necessary queries for the failure of the software. It took seven hours to fix.

Case 02 solution: We can see that in the case-2 scenario a user confirmed a product by the development team about malfunction in the system. As the software became hugely complex, this is called the “Interactive Error Offer delivery.”

INDEX