DATE: / /

PAGE NO:

EXPERIMENT 1

Aim: To determine the nature of roots of a quadratic equation, its input is triple of positive integers (say x, y, z) and values may be from interval [1, 100] the program Perform Boundary value and Ro... Analysis.

Theory

Values where we test variables

min value : 1 — value preceding min : 0

max value : 100 — value succeeding max : 101

value succeeding min : 2 — value preceding max : 99

nominal values : 50

Totals test cases BVA : 4n+1

Robust : 6n+1

Quadratic roots will be

-> real if $b^2 - 4ac > 0$

-> imaginary if $b^2 - 4ac < 0$

-> equal if $b^2 - 4ac = 0$

-> not quadratic if $a = 0$

Boundary Value Test Cases

Test Case	a	b	c	Expected Output
1	1	50	50	Real roots
2	100	50	50	Imaginary roots
3	2	50	50	real roots
4	99	50	50	Imaginary roots
5	50	50	50	Imaginary roots
6	50	1	50	Imaginary roots
7	50	2	50	Imaginary roots

Sign.....

DATE: / /
PAGE NO:

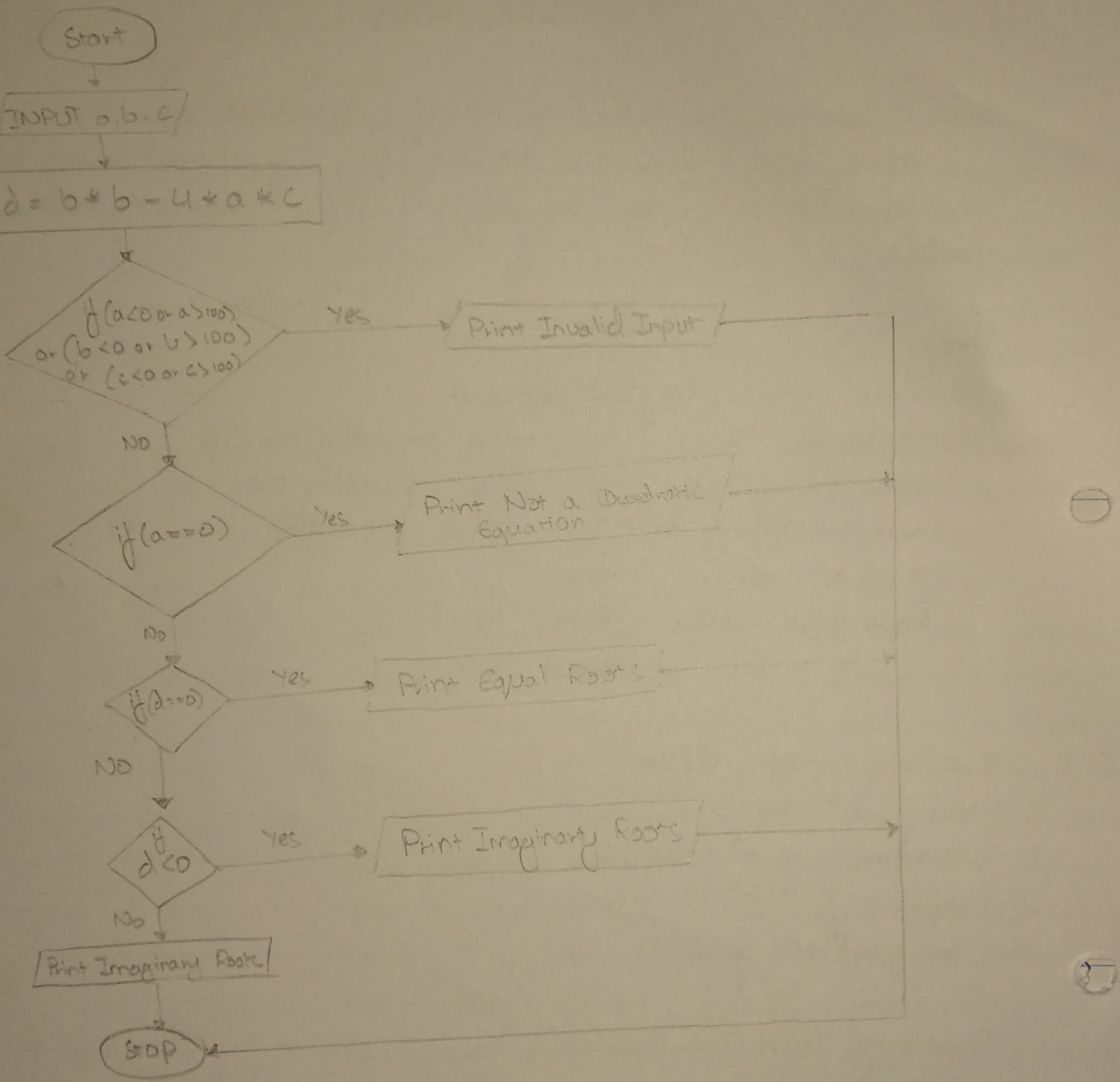
8	50	100	50	Equal Roots
9	50	99	50	Imaginary Roots
10	50	50	1	Real Roots
11	50	50	100	Imaginary Roots
12	50	50	2	Real Roots
13	50	50	99	Imaginary Roots

Robust Test Cases

Test Case	a	b	c	Expected Outcome
1	0	50	50	Not Quadratic
2	1	50	50	Real Roots
3	2	50	50	Real Roots
4	50	50	50	Imaginary Roots
5	94	50	50	Imaginary Roots
6	100	50	50	Imaginary Roots
7	101	50	50	Input Value out of domain
8	50	0	50	Imaginary Roots
9	50	1	50	Imaginary Roots
10	50	2	50	Imaginary Roots
11	50	99	50	Imaginary Roots
12	50	100	50	Equal Roots
13	50	101	50	Input Value out of domain
14	50	50	0	Real Roots
15	50	50	1	Real Roots
16	50	50	2	Real Roots
17	50	50	99	Imaginary Roots
18	50	50	100	Imaginary Roots
19	50	50	101	Input Value out of domain

Sign

FLOWCHART


Start
INPUT a,b,c
d = b*b - 4*a*c
if(a<0 or a>100) or (b<0 or b>100) or (c<0 or c>100)
Yes
Print Invalid Input
NO
if(a==0)
Yes
Print Not a Quadratic Equation
No
if(d==0)
Yes
Print Equal Roots
NO
if d<0
Yes
Print Imaginary Roots
No
Print Imaginary Roots
STOP

Program:

```
#include<iostream>
using namespace std;
void find_roots(int a, int b, int c) {
     if (a == 0) {
        cout << "\nNot Quadratic";
        return;
     }
     int d  = b*b - 4*a*c;
     if (d > 0) { cout << "\nReal Roots";}
     else if (d == 0) { cout << "\nEqual Roots";}
     else { cout << "\nImaginary Roots";}
}

int main() {
    int a,b,c;
    cout<<"Enter the the value of a,b,c for quadratic equation ax^2 + bx +
c:";
    cin>>a>>b>>c;
    if(a>100||b>100||c>100){cout<<"\n Input value out of domain";}
    else if(a<0||b<0||c<0){cout<<"\n Input value out of domain";}

    find_roots(a, b, c);
    return 0;
}
```

Output:

```
kunal@Kunal-Notebook:~/Documents/CollegeSem7/STQA/Practicals/Pract1
-$ g++ SQTA_Practical_1.cpp
kunal@Kunal-Notebook:~/Documents/CollegeSem7/STQA/Practicals/Pract1
-$ ./a.out
Enter the the value of a,b,c for quadratic equation ax^2 + bx + c:50 100 50

Equal Rootskunal@Kunal-Notebook:~/Documents/CollegeSem7/STQA/Practicals/Pract1
-$
```

DATE: / /

PAGE NO:

EXPERIMENT 2

Aim: To determine the type of triangle. It input is triple of positive integer (say x, y, z) and the values may be from interval [1, 100]. The program output may be one of the following [Scalene, Isosceles, Equilateral, Not a triangle]. Perform BVA and Robust Case Testing.

Theory

Triangle problem accepts three positive integers a, b, c are three sides of the triangle. For a, b, c to form a triangle the following condition must be satisfied: $a < b+c$, $b < a+c$, $c < a+b$ and also $a > 0$, $b > 0$, $c > 0$ if any of these conditions are not satisfied the resulting output is not a triangle.

i) For equilateral Δ, $a = b = c$

ii) For Isosceles Δ, $a = b$ or $b = c$ or $c = a$

Boundary Value Analysis

Test Case	a	b	c	EXPECTED OUTPUT
1	1	50	50	Isosceles Triangle
2	2	50	50	Isosceles Triangle
3	50	50	50	Equilateral Triangle
4	99	50	50	Isosceles Triangle
5	100	50	50	Not a Triangle
6	50	1	50	Isosceles Triangle
7	50	2	50	Isosceles Triangle
8	50	99	50	Isosceles Triangle
9	50	100	50	Not a Triangle
10	50	50	1	Isosceles Triangle
11	50	50	2	Isosceles Triangle
12	50	50	99	Isosceles Triangle
13	50	50	100	Not a Triangle

Sign

DATE: / /

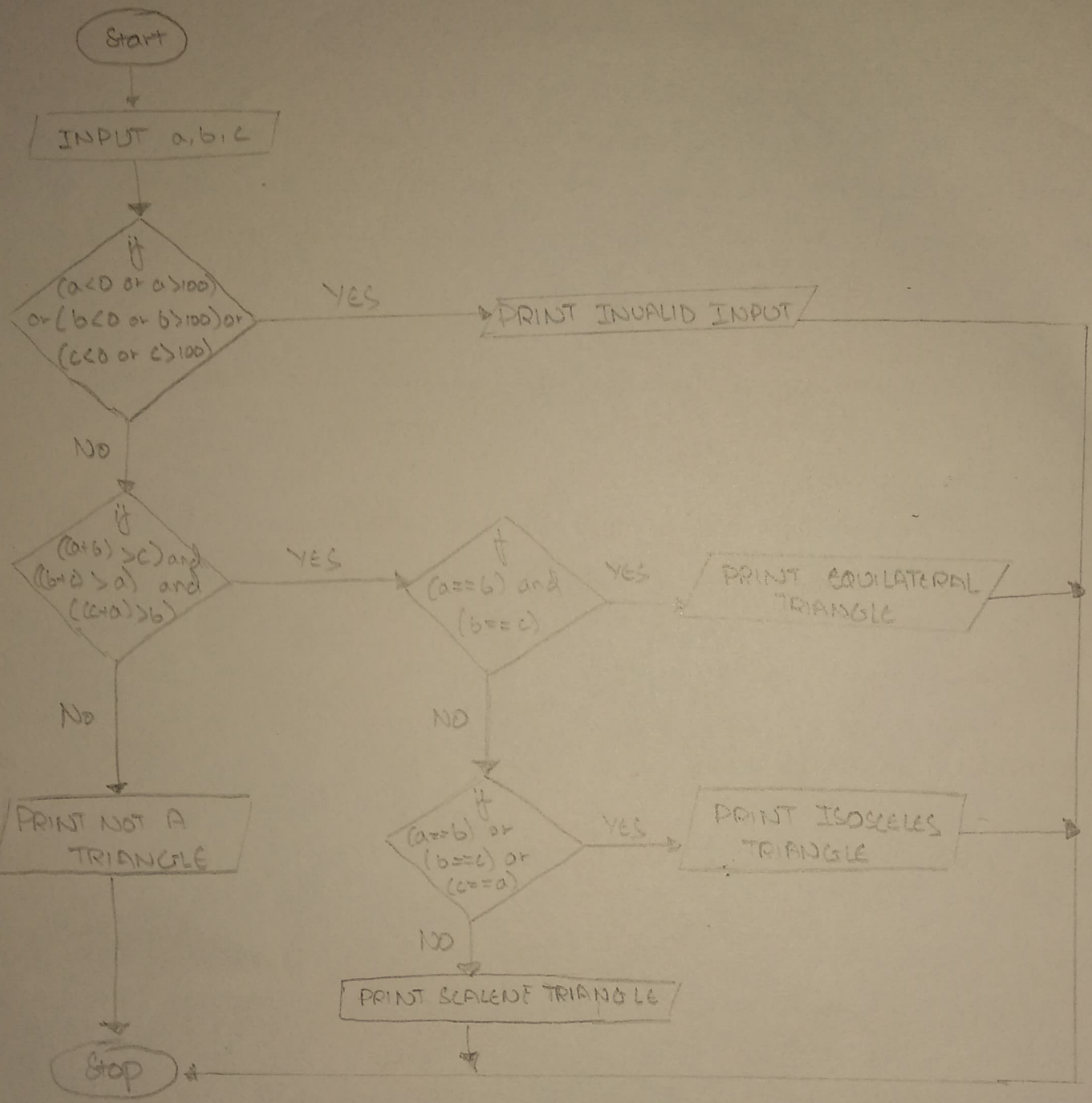
PAGE NO:

Robust Test Cases

Test Case	a	b	c	Expected Outcome
1	0	50	50	Invalid Input
2	1	50	50	Isosceles Triangle
3	2	50	50	Isosceles Triangle
4	50	50	50	Equilateral Triangle
5	99	50	50	Not a Triangle
6	100	50	50	Not a Triangle
7	101	50	50	Invalid Input
8	50	0	50	Invalid Input
9	50	1	50	Isosceles Triangle
10	50	2	50	Isosceles Triangle
11	50	99	50	Isosceles Triangle
12	50	100	50	Not a Triangle
13	50	101	50	Invalid Input
14	50	50	0	Invalid Input
15	50	50	1	Isosceles Triangle
16	50	50	2	Isosceles Triangle
17	50	50	99	Isosceles Triangle
18	50	50	100	Not a Triangle
19	50	50	50	Invalid Input

Sign.....

FLOWCHART


Start
INPUT a,b,c
if (a<0 or a>100) or (b<0 or b>100) or (c<0 or c>100)
YES
PRINT INVALID INPUT
NO
if (a+b)>c) and (b+c>a) and (c+a)>b)
YES
if (a==b) and (b==c)
YES
PRINT EQUILATERAL TRIANGLE
No
NO
PRINT NOT A TRIANGLE
if (a==b) or (b==c) or (c==a)
YES
PRINT ISOSCELES TRIANGLE
NO
PRINT SCALENE TRIANGLE
Stop

Code :

```
#include<iostream>
#include<bits/stdc++.h>

using namespace std;


void find_triangle(int a, int b, int c) {
    if (((a+b)>c)&&((b+c)>a)&&((c+a)>b)){
        if ((a==b)&&(b==c)) {
            cout << "\nEquilateral Triangle";
        }
        else if ((a==b)||(b==c)||(c==a)){
            cout << "\nIsosceles Triangle";
        }
        else{
            cout << "\nScalene Triangle";
        }
    }
    else{
        cout<<"\nNot a Triangle";
    }
}

int main() {
    int a,b,c;
    cout<<"Enter the the value of a,b,c respectively: ";
    cin>>a>>b>>c;
    if(a>100||b>100||c>100||a<0||b<0||c<0){
        cout<<"\nInput value out of domain. Invalid Input!";
    }
    find_triangle(a, b, c);
    return 0;
}
```

Output :

```
kunal@Kunal-Notebook:~/Documents/CollegeSem7/STQA/Practicals/Pract2
–$ g++ pract2.cpp
kunal@Kunal-Notebook:~/Documents/CollegeSem7/STQA/Practicals/Pract2
–$ ./a.out
Enter the the value of a,b,c respectively: 50 50 50

Equilateral Trianglekunal@Kunal-Notebook:~/Documents/CollegeSem7/STQA/Practicals/Pract2
–$
```

DATE: / /

PAGE NO:

EXPERIMENT 3

Aim: Consider the program of classification of triangle. Its input is a triple of positive integers (say a,b,c) and values for each of these may be from interval [1,100]. The output may have one of the options given below:
Obtuse-Angled, Acute-Angled, Right-angled or invalid triangle or input values out of range. Find all the du-paths & identify those du-paths that are definition clear. Also find all du-paths, all uses and all definitions & generate test cases for them.

Solution

Variable	Define At Node	Used At Node
a	8	13,14,22,23,24
b	10	13,14,22,23,24
c	12	13,14, 22-24
a1	22	25,28
a2	23	25,28
a3	24	25,28
valid	5,15,18	21,35

CODE

```
#include <IOSTREAM>
using namespace std;
 int main(){
        double a,b,c, a1,a2,a3;
        int valid=0;
        cout << "Enter first side of the triangle:";
        cin>a;
        cout << "\n Enter 2nd side of the triangle:";
        cin>b;
        cout << "\n Enter 3rd side of the triangle:";
```

Sign.....

Be Positive

DATE: / /
PAGE NO:

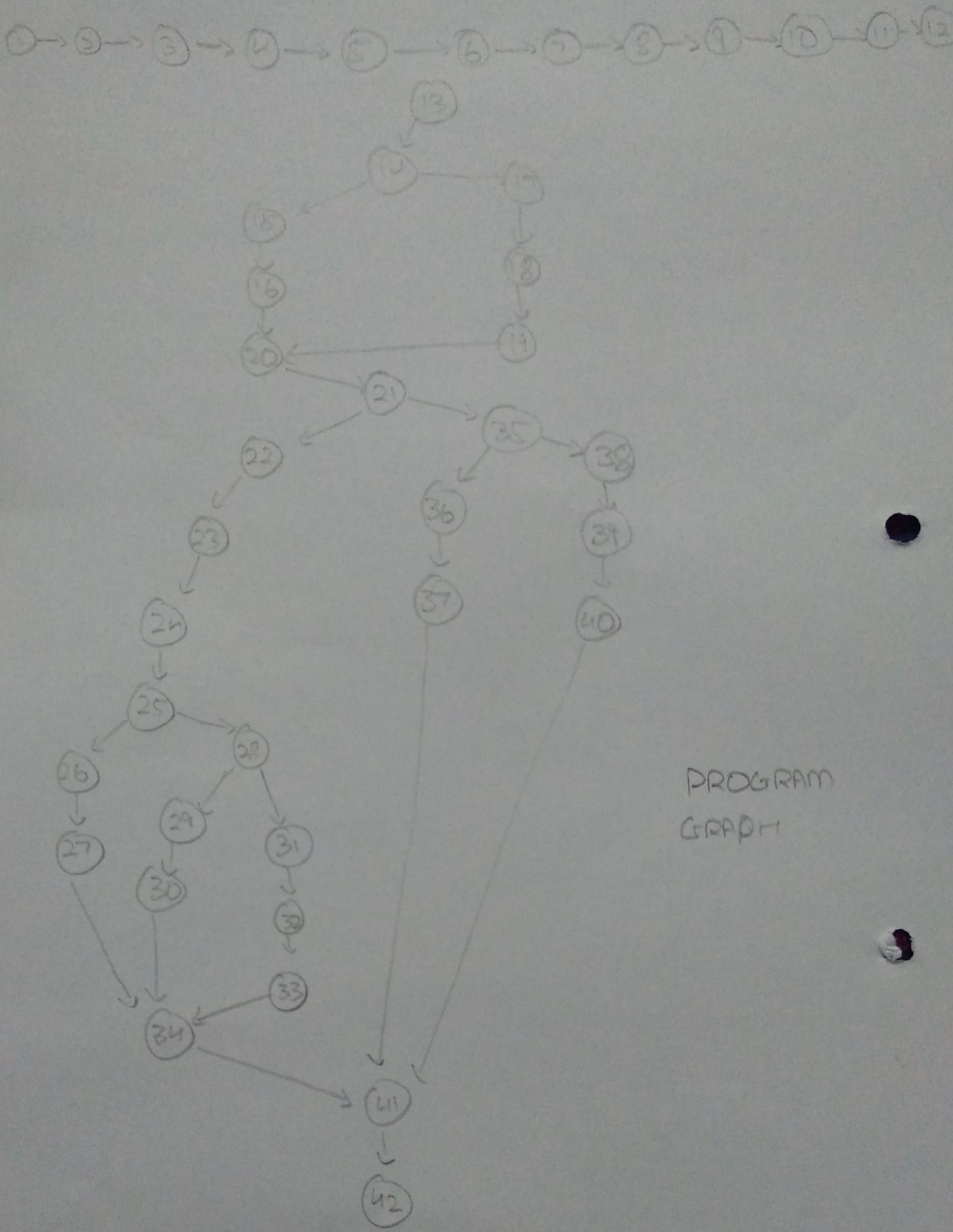
```
cin>>c;
if(a>0 && a<=100 && b>0 && b<=100 && c>0 && c<=100)
    if((a+b)>c && (b+c)>a && (a+c)>b){
        valid = 1;
    }else{
        valid = -1;
    }
if(valid==1){
    a1 = (a*a + b*b)/(c*c);
    a2 = (b*b + c*c)/(a*a);
    a3 = (c*c + a*a)/(b*b);
    if(a1<1 || a2<1 || a3<1)
        cout<<"\nObtuse Angled Triangle";
    else if(a1==1 || a2==1 || a3==1)
        cout<<"\nRight Angled Triangle"
    else
        cout<<"\nAcute Angled Triangle";
}
else if(valid==-1)
    cout<<"\nInvalid triangle";
else
    cout<<"\nInputs values out of range";
return 0;
}
```

All du paths:

All du paths	Definition Clear	All du paths	Definition Clear
8-13	Yes	8-14,17-22	Yes
8-14	Yes	8-13,21,22	Yes
8-16,20-22	Yes	8-16, 20-23	Yes

Be Positive

Sign


PROGRAM
GRAPH

DATE / /
PAGE NO

All du paths	Definition Clear
8-14, 17-23	Yes
8-13, 21-23	Yes
8-16, 20-24	Yes
8-14, 17-24	Yes
8-13, 21-24	Yes
12-14, 17-22	Yes
12,13,21,22	Yes
12,13	Yes
12-16,20-23	Yes
12-14, 17-23	Yes
12,13, 21-23	Yes
12-16, 20-24	Yes
12-14, 17-24	Yes
12,13,21-24	Yes
22-25	Yes
22-25, 28	Yes
23-25	Yes

All du paths	Definition Clear
10-13	Yes
10-14	Yes
10-16, 20-22	Yes
10-14, 17-22	Yes
10-13, 21, 22	Yes
10-16, 20-23	Yes
10-14, 17-23	Yes
12-14	Yes
12-16, 20-22	Yes
23-25, 28	Yes
24, 25	Yes
24, 25, 28	Yes
5-16, 20, 21	No
5-14, 17-21	No
5-13, 21	Yes
5-16, 20, 21, 35	No
5-14, 17-26, 35	No

All du paths	Definition Clear
8-13, 21, 25	Yes
15, 16, 20, 21	Yes
15, 16, 20, 21	Yes
18-21	Yes
18-21, 35	Yes

All Uses Paths:

All Uses	Definition Clear
8-13	Yes
8-14	Yes
8-16, 20-22	Yes

All Uses	Definition Clear
23-25, 28	Yes
24, 25	Yes
24, 25, 28	Yes

Sign.....

DATE: / /

PAGE NO:

All Uses	Definition Clear
8-16, 20-23	Yes
8-16, 20-24	Yes
10-13	Yes
10-14	Yes
10-16, 20-24	Yes
10-13, 21-23	Yes
10-16, 20-24	Yes
12, 13	Yes
12-16, 20-24	Yes
22-25	Yes
23-25	Yes

All Uses	Definition Clear
5-16, 20, 21	No
5-14, 17-21, 35	No
15, 16, 20, 21	Yes
15, 16, 20, 21, 35	Yes
12-14	Yes
12-16, 20, 21, 22	Yes
12-16, 20-23	Yes
18, 21	Yes
18-21, 35	Yes
22-25, 28	Yes

Test Cases for All Use cases:

	a	b	c	Expected Output	Remarks
1	30	20	40	Obtuse	8-13
2	30	20	40	Obtuse	8-14
3	30	20	40	Obtuse	8-16, 20-22
4	30	20	40	Obtuse	8-16, 20-23
5	30	20	40	Obtuse	8-16, 20-24
6	30	20	40	Obtuse	10-13
7	30	20	40	Obtuse	10-14
8	30	20	40	Obtuse	10-16, 20-22
9	30	20	40	Obtuse	10-13, 21-23
10	30	20	40	Obtuse	10-16, 20-24
11	30	20	40	Obtuse	12, 13
12	30	20	40	Obtuse	12-14
13	30	20	40	Obtuse	12-16, 20, 21, 22
14	30	20	40	Obtuse	12-16, 20-23
15	30	20	40	Obtuse	12-16, 20-24

Sign

DATE: / /
PAGE NO:

16	30	20	40	Obtuse	22-25
17	30	40	50	Obtuse	22-25, 28
18	30	20	40	Obtuse	23-25
19	30	40	50	Right	23-25, 28
20	30	20	40	Obtuse	24, 25
21	30	40	50	Right	24, 25, 28
22	30	20	40	Obtuse	5-16, 20, 21,
23	-	-	-	-	5-16, 20, 21, 35
24	30	10	15	Invalid	18-21
25	30	10	15	Invalid	18-21, 35

Sign.....

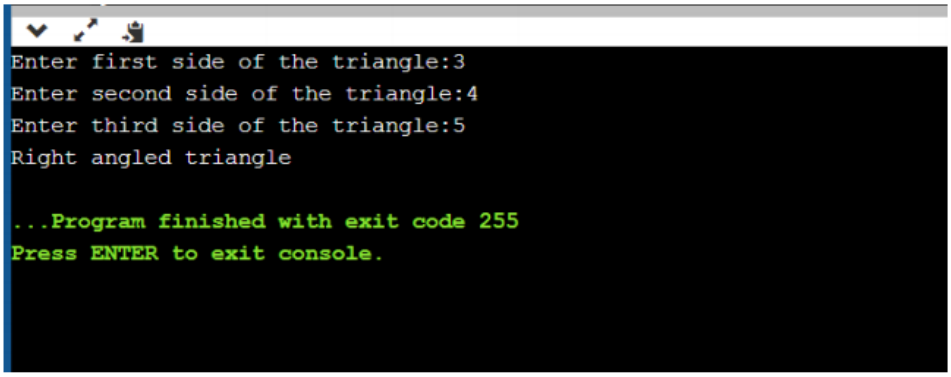
SNO.	a	b	c	EXPECTED OUTPUT	REMARKS
1	30	20	40	Obtuse Angled Triangle	8-13
2	30	20	40	Obtuse	8-14
3	30	20	40	Obtuse	8-16, 20-22
4	-	-	-	-	8-14, 17-22
5	-	-	-	-	8-13, 21, 22
6	30	20	40	Obtuse	8-16, 20-23
7	-	-	-	-	8-14, 17-23
8	-	-	-	-	8-13, 21-23
9	30	20	40	Obtuse	8-16, 20-24
10	-	-	-	-	8-14, 17-24
11	-	-	-	-	8-13, 21-24
12	30	20	40	Obtuse	10-13
13	30	20	40	Obtuse	10-14
14	30	20	40	Obtuse	10-16, 20-22
15	-	-	-	-	10-14, 17-22
16	-	-	-	-	10-14, 17-23
17	30	20	40	Obtuse	10-16, 20-23
18	-	-	-	-	10-14, 17-24
19	-	-	-	-	10-13, 21-24
20	30	20	40	Obtuse	10-16, 20-24
21	-	-	-	-	10-14, 17-24
22	-	-	-	-	10-13, 21-24
23	30	20	40	Obtuse	12, 13
24	30	20	40	Obtuse	12-14
25	30	20	40	Obtuse	12-16, 20-22
26	-	-	-	-	12-16, 20-22
27	-	-	-	-	12, 13, 21-22
28	30	20	40	Obtuse	12-16, 20-23
29	-	-	-	-	12-14, 17-23
30	-	-	-	-	12, 13, 21-23

S.No.	a	b	c	Expected Output	Remarks
31	30	20	40	Obtuse	12-16, 20-24
32	-	-	-	-	12-14, 17-24
33	-	-	-	-	12, 13, 21-24
34	30	20	40	Obtuse	22-25
35	30	40	50	Right	23-25, 28
36	30	~~20~~	40	Obtuse	24, 25
37	30	40	50	Right	23-25, 28
38	30	20	40	Obtuse	24, 25
39	30	40	50	Right	24, 25, 28
40	30	20	40	Obtuse	5-16, 20, 21
41	30	10	15	Invalid	5-14, 17-21
42	102	-1	6	Input values out of range	5-13, 21, 35
43	-	-	-	-	5-16, 20, 21, 35
44	30	10	15	Invalid Triangle	5-14, 17-21, 35
45	102	-1	6	Input values out of range	5-13, 21, 35
46	30	20	40	Obtuse	15, 16, 20, 21
47	-	-	-	-	15, 16, 20, 21, 35
48	30	10	15	Invalid Triangle	18-21
49	30	10	15	Invalid Triangle	18-21, 39

Test cases for All definitions

Test	a	b	c	Expected Output	Remarks
1	30	20	40	Obtuse Angled Triangle	8-13
2	30	20	40	Obtuse	10-13
3	30	20	40	Obtuse	12, 13
4	30	20	40	Obtuse	22-25
5	30	20	40	Obtuse	23-25
6	30	20	40	Obtuse	24-25
7	30	20	40	Obtuse	5-16, 20, 21
8	30	20	40	Obtuse	5-16, 20, 21
9	30	10	13	Invalid Triangle	18-21

OUTPUT:


Enter first side of the triangle:3
Enter second side of the triangle:4
Enter third side of the triangle:5
Right angled triangle
...Program finished with exit code 255
Press ENTER to exit console.

DATE / /
PAGE NO

EXPERIMENT 11

Aim: Consider an automating banking application. The user can dial the bank from a PC, provide a 6 digit password and follow with a series of keyword commands that activate the banking function. The software for the application accepts that data in the form of

Area Code: Blank | 3 digit Number
Prefix : 3 digit number not beginning with 0 or 1
Suffix : 4 - digit number
Password: 6 character alphanumeric
Commands: check status, deposit, withdrawal
Design adhoc test cases to test the system

Theory

Adhoc testing is an informal or unstructured software testing type that aims to break the testing process in order to find possible defects or error at any an early possible stage.
Adhoc testing is done randomly and it is usually an unplanned activity which doesn't follow any documentation.

CODE

```
public class Banking {
    static int balance = 0;
    public static void check() {
        System.out.println("Current Balance = " + balance);
    }
    public static void withdraw() {
        System.out.println("Enter amount to be withdrawn:");
        Scanner sc = new Scanner(System.in);
        int wd = sc.nextInt();
        if (balance < wd) { System.out.println("Not enough balance"
```

Be Positive

Sign

DATE: / /
PAGE NO:

```
        else {
            balance -= wd ; System.out.println("Withdrawn Sucessfully
        }
    }
    public static void deposit(){
        System.out.println("Enter deposit amount:");
        Scanner sc = new Scanner(System.in)
        int dep = sc.nextInt();
        if(dep>=0){
            balance += dep;
        } else{
            System.out.println("Enter amount:")
        }
    }

    public static void main(String[] args) {
        long pass = 123456
        String alphapass = "";
        Scanner sc = new Scanner(System.in);
        String abc = sc.nextLine();
        Long upass = Long.parseLong(abc)
        if(pass != upass) { return; }
        String areacode = sc.nextline()
        if(areacode.equals("") || areacode.length==3){
            System.out.println("Invalid Area Code"); return;
        }
        String pre = sc.nextline();
        int prefix = Integer.parseInt(pre);
        if(!(prefix/100)>1 && prefix/1000 ==0) {
            System.out.println("Invalid Prefix");
            return;
        }
```


Sign.....

DATE
PAGE NO.

```
System.out.println ("Enter 4digit Suffix: ");
String suffix = sc.nextline();
if(suffix.length() != 4) {
        System.out.println ("Invalid Suffix")
        return;
}
alphapass = sc.nextline();
if (alphapass.length() != 6){
    System.out.println ("Invalid Password");
    return;
} else if(alphapass.matches ("[A-Za-z0-9]+") == false){
    System.out.println ("Password not alphanumeric")
    return;
}
char ch;
do {
  System.out.println("Select \n 1CheckStatus \n 2 Deposit
                        +"\n3 Withdraw \n");
  int command = sc.nextInt();
  switch(command) {
    case 1: check(); break;
    case2 : deposit(); break;
    case 3; withdraw(); break;
  }
  System.out.print ("Doyou wish to continue ? ")
  ch=sc.next().charAt(0);
} while (ch=='Y'||ch=='y');
    }
}
```

Result Successfully designed ADHOC test cases for testing.

Sign.....

INPUT					EXPECTED OUTPUT	ACTUAL OUTPUT	Result
AREA CODE	Prefix	Suffix	Pass.	Comm			
123	222	2323	sd@es	checkt	INVALID	INVALID	INVALID
123	122	3232	3ejrcsj	with	INVALID	INVALID	INVALID
	221	323	d2d2d3	Dep.	INVALID	INVALID	INVALID
123	011	3332	dfdIs	checkt.	INVALID	INVALID	INVALID
123	4567	2232	Ug73f	with.	INVALID	INVALID	INVALID
123	234	12ab	sdssed	with.	VALID	VALID	VALID
	203	3562	Ask45	Dep.	VALID	VALID	VALID
321	657	7865	que34	With.	VALID	VALID	VALID
	345	6875	pasund	ch.s.	VALID	VALID	VALID

DATE: / /
PAGE NO:

EXPERIMENT 5

Aim: Prepare a test plan document for library management system.

Test plan: Library Management System

Table of Contents:

1	Introduction	1
2	Purpose	1
3	Scope	2
4	Testing Strategies	2
	4.1 Unit Testing	2
	4.2 System and Integration Testing	3
	4.3 Performance and Stress Testing	3
	4.4 Accepting Testing	3
5	Features to be Tested	3
6	Hardware Requirement	4
7	Environment Requirement	4
8	Test Schedule	4
9	Risk and Mitigation	5
10	Tools	5

Sign.....

DELTA Pg No.
Date / /

Introduction

Every college has their library both for teachers & students use. The traditional system to manage them is either keeping tracks of them in a register or keeping track of a similar entry in computer in excel format. This working is fine until you need data from the logs.

Purpose

The library management system is an online application for existing a librarian in managing a book library in a University. The experiment (system) should provide a basic set of features to add/update books, search for books & manage them

This test plan document supports following objectives:

- → Identify existing project information & software that should be tested
- → List of reccomended test requirements
- → Description of testing techniques to be employed
- → Identify required resources & provide an estimate of test efforts
- → list the deliverables elements of test activities.

Scope

The system that is to be developed provides the related information on students and system administrator.

→ Creating a system admin who will manage system
→ Admin will have login activity.
→ Can add/edit/delete/view the category
→ Can add/edit/delete/view the authors
→ Can add/edit/delete view the books
→ Can add/edit/delete/view books issued
→ Can search for books.

Testing Strategies

Unit Testing

In order to test a single module, we need to provide a complete environment i.e. besides the module we would requires:

→ procedures belonging to other modules
→ Non local data structures
→ procedure to call the functions of module.

Unit Testing was done on each & every module:

Test for admin module:

(i) Testing login Interface
(ii) Student account addition
(iii) Book addition

2. Test for student login module
   (i) Test for Student login interface
   (ii) Test for account creation

System & Integration Testing
UI: interface from where all inputs are given
DBMS: place where all calculations are done & stored
VAL: Validation Module
CNI: These contents are displayed in the reports.

Performance & Stress Testing:
Stress testing involving testing beyond normal operational opacity. Normally this kind of test is done to determine the system's robustness in terms of extreme load for example multi-user testing.

User Acceptance Testing
There are different types of acceptance testing. Most Common is user acceptance testing.
Features to be Tested:
→ GUI tested
→ Database Test
→ Base Operations Test
→ Advanced Operations Test

DELTA Pg No. Date / /

(i) Test for Student login Module → Check in & Check out test
(ii) Test for Student login interface → Bill
(iii) Test for account creation

~~System & Integration~~ Hardware Requirement:
→ Os: Windows → IDE: VS
→ Language: Python → Browser: Chromium.
→ Database: MySQL

Environment Requirement
Mainframe: Specify both necessary and desired properties of the test environment.
Workstation: Computers in library to be used
Mobile Phones through which records can be viewed

Test Schedule:

SNo.	Task	Days	Start Time	End Time	Responsibility
1	Understanding & Analysing Reports	5	2 July	7 July	Team
2	Generate Test Scenarios	10	7 July	17 July	Member 1
3	Test Case Documentation	40	7 July	17 Aug	Member 2
4	Verify env. setup	01	17 July	17 Aug	Member 3
5	Unit Testing	10	18 Aug	28 Aug	Member 4
6	Integration Testing	10	28 Aug	7 Sept	Member C
7	Final Testing	15	7 Sept	22 Sept	End User
8	Summary Report	1	23 Sept	23 Sept	Team

DELTA Pg No.
Date / /

Risk & Mitigation

(i) Reassigned Developer, staff in limited staff to avoid pending work

(ii) Keeping a backup of ~~generating~~ battery for electricity issues

Tools:

-> Selenium

-> QTP

DATE
PAGE NO

EXPERIMENT 6

Aim: study of any Testing Tool (Win Runner)

Theory: Win Runner is a program that is responsible for the automated testing of software.

Importance of automated testing:

i) Reduced testing time
ii) Consistent Test procedures
iii) Eliminates errors of manual testing
iv) Improved testing productivity

Uses of Win Runner

i) Using Win Runner sophisticated automated tests can be created and run on application
ii) A series of wizards will be provided to the user, and these wizard can create tests in an automated manner
iii) The goal of Winrunner is to make sure business processes are properly carried out. Winrunner uses test script language.

Win Runner Testing Process

i) Create the GUI Map: The first stage is to create the GUI map in Win Runner can recognize the GUI Objects in the application being tested.

ii) Create Tests: Next is creation of test scripts by recording, programming, or a combination of both during this process, Win Runner captures data and saves it as expected results.

iii) Debug Tests: Run Tests in debug mode, to make sure they run smoothly. One can set breakpoints, monitor

iv) Run Tests: Tests can be run in verify mode to test the appliation

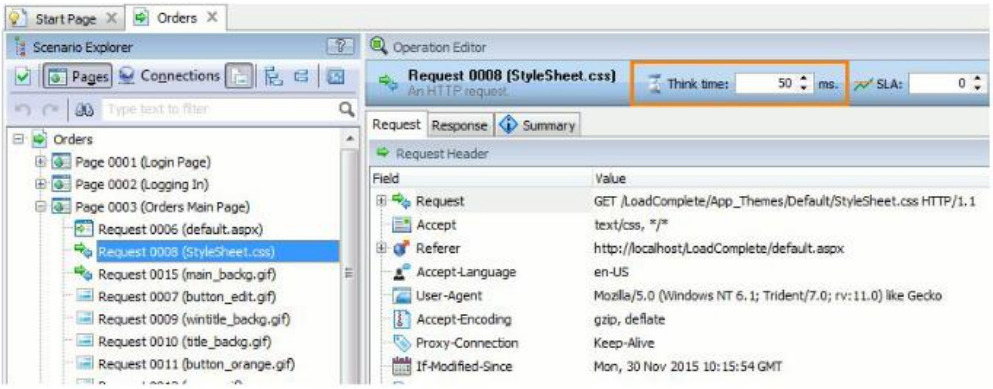
Sign.....

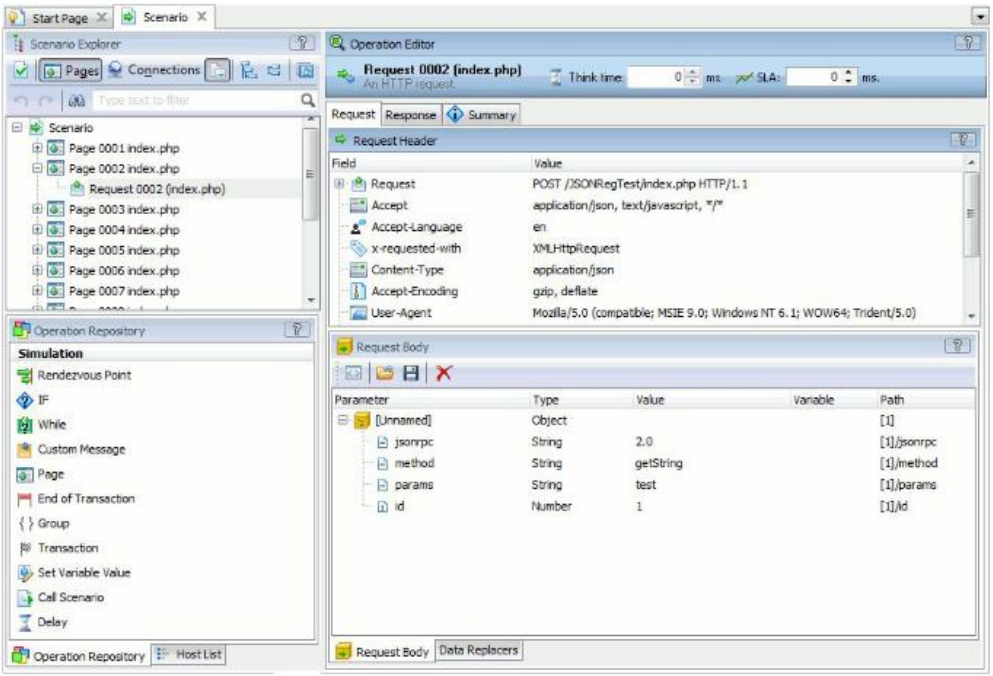
DATE: / /
PAGE NO:

v) View Results : Following each test run, win runner displays the results in a report. We can view results in the standard win runner report view. The winrunner report viewse displays the test results in a windows-style view.

vi) Report Defects: If a test run fails due to a defect in the application being tested, one can ~~defect~~ report information about the defect directly from the test results window.

Sign.....


Start Page
Orders
Scenario Explorer
Pages
Connections
Type text to filter
Orders
Page 0001 (Login Page)
Page 0002 (Logging In)
Page 0003 (Orders Main Page)
Request 0006 (default.aspx)
Request 0008 (StyleSheet.css)
Request 0015 (main_backg.gif)
Request 0007 (button_edit.gif)
Request 0009 (wintitle_backg.gif)
Request 0010 (title_backg.gif)
Request 0011 (button_orange.gif)
Operation Editor
Request 0008 (StyleSheet.css)
An HTTP request
Think time: 50 ms.
SLA: 0
Request
Response
Summary
Request Header
Field
Value
Request
GET /LoadComplete/App_Themes/Default/StyleSheet.css HTTP/1.1
Accept
text/css, */*
Referer
http://localhost/LoadComplete/default.aspx
Accept-Language
en-US
User-Agent
Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko
Accept-Encoding
gzip, deflate
Proxy-Connection
Keep-Alive
If-Modified-Since
Mon, 30 Nov 2015 10:15:54 GMT


Start Page
Scenario
Scenario Explorer
Pages
Connections
Type text to filter
Scenario
Page 0001 index.php
Page 0002 index.php
Request 0002 (index.php)
Page 0003 index.php
Page 0004 index.php
Page 0005 index.php
Page 0006 index.php
Page 0007 index.php
Operation Repository
Simulation
Rendezvous Point
IF
While
Custom Message
Page
End of Transaction
Group
Transaction
Set Variable Value
Call Scenario
Delay
Operation Repository
Host List
Operation Editor
Request 0002 (index.php)
An HTTP request
Think time: 0 ms.
SLA: 0 ms.
Request
Response
Summary
Request Header
Field
Value
Request
POST /JSONRegTest/index.php HTTP/1.1
Accept
application/json, text/javascript, */*
Accept-Language
en
x-requested-with
XMLHttpRequest
Content-Type
application/json
Accept-Encoding
gzip, deflate
User-Agent
Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0)
Request Body
Parameter
Type
Value
Variable
Path
[Unnamed]
Object
[1]
jsonrpc
String
2.0
[1]/jsonrpc
method
String
getString
[1]/method
params
String
test
[1]/params
id
Number
1
[1]/id
Request Body
Data Replacers

DATE / /
PAGE NO:

EXPERIMENT 7

Aim: Study of any web testing tool (selenium)

Theory: Selenium is a robust set of tools that supports rapid development of test automation for web based applications.

Selenium Components:

i) Selenium is composed of three major tools.

ii) Selenium - RC provides an API and library for each of its supported languages: HTML, Java, C#, Perl, Python and Ruby.

Selenium Grid: It allows the selenium - RC solution to scale for large test suites that must run in multiple environments. With Selenium - Grid, multiple instances of Selenium - RC are running on various operating system and browser configurations: each of these when launching register with a hub.

Flexibility and Extensibility

Selenium is highly flexible. Selenium IDE allows for the addition of user - defined - user - extensions for creating additional commands customized to the user's needs.

Test Suite: A test suite is a collection of tests often one will run all the tests in a test suite as one continuous batch job. When using Selenium IDE, test suites also can be defined using a simple HTML file.

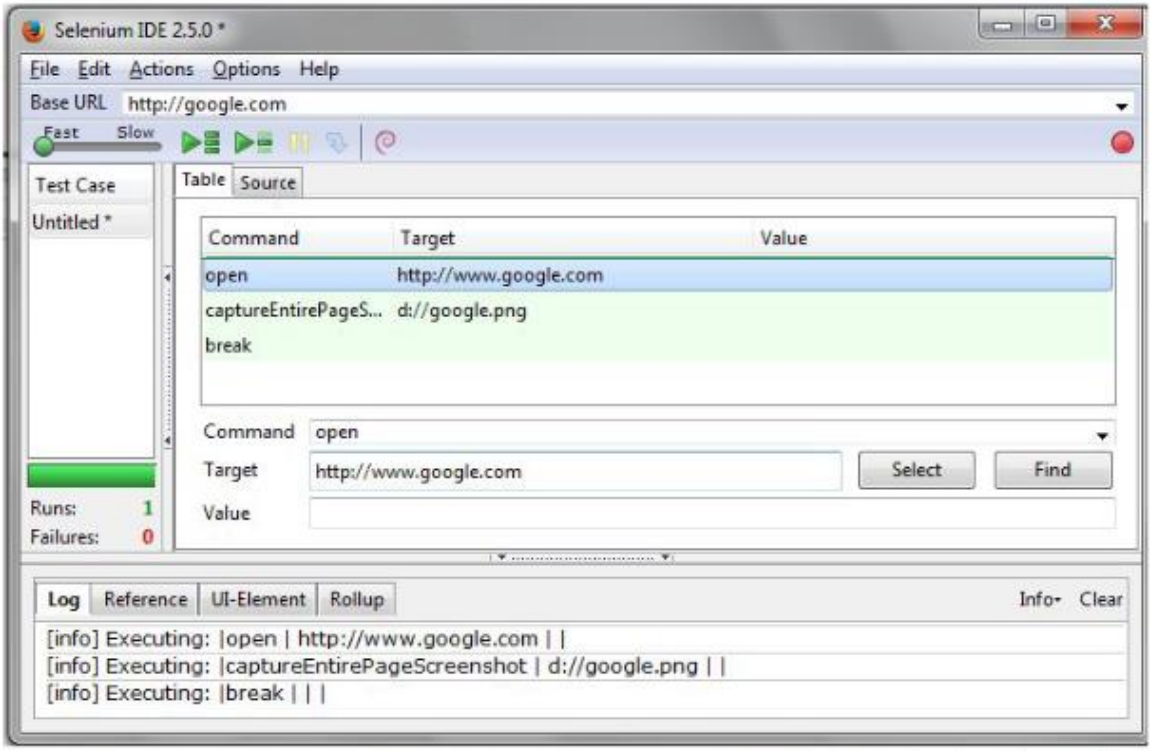
Test suites can also be maintained when using Selenium RC via programming. Few typical Selenium commands are: —

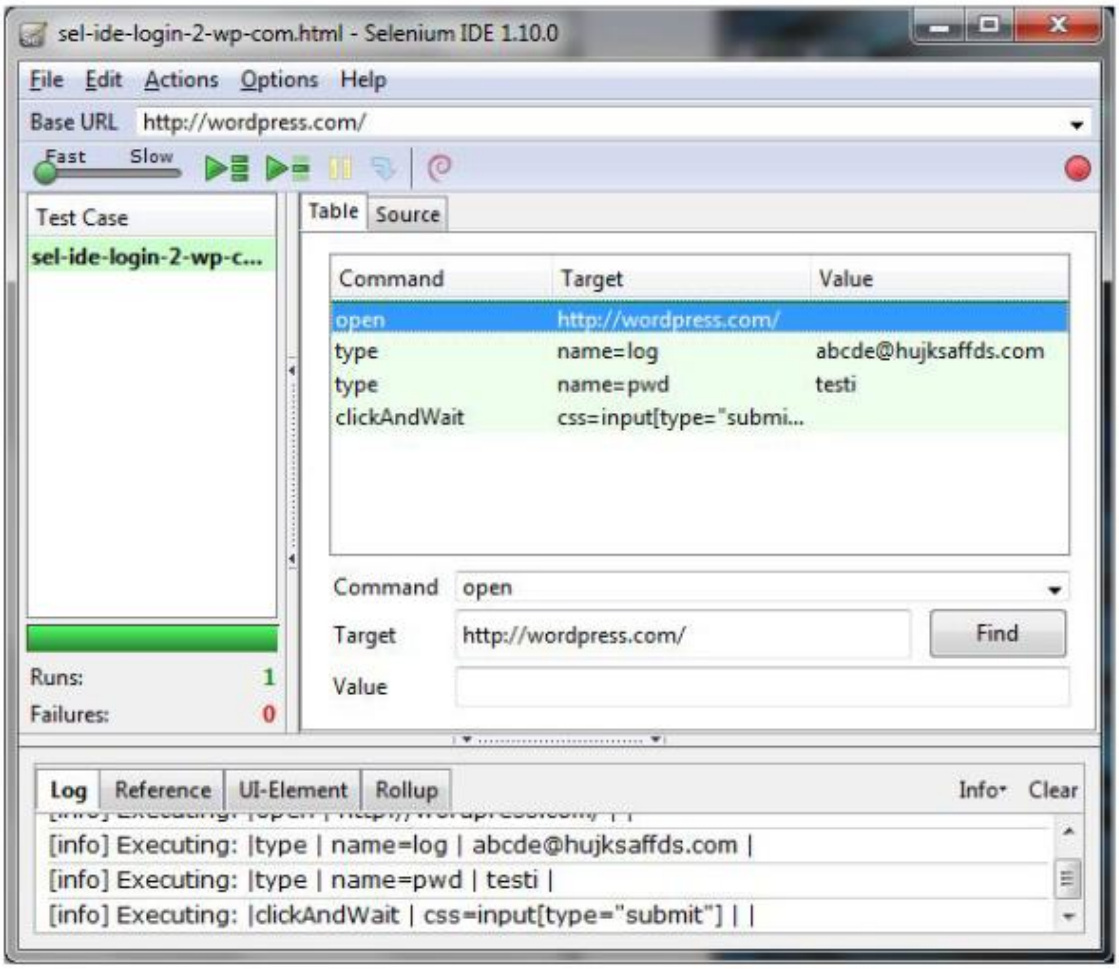
Sign.....

DATE
PAGE NO

i) open: opens a page using a URL
ii) click / clickAndWait: Performs a click operation, and optionally waits for a new page load.
iii) VerifyTable: Verifies a table's expected contents
iv) verifyTitle / assertTitle: Verifies an expected page title
v) verifyTextPresent: Verifies expected text is somewhere on this page
vi) waitForPageToLoad: Pauses execution until an expected new page load
vii) waitForElementPresent: Pauses execution until an expected UI as defined by its HTML Tag, is present on the page.

Result: Successfully studied web testing tool (selenium).


Selenium IDE 2.5.0 *
File Edit Actions Options Help
Base URL http://google.com
Fast Slow
Test Case
Untitled *
Table Source
Command Target Value
open http://www.google.com
captureEntirePageS... d://google.png
break
Command open
Target http://www.google.com
Select
Find
Value
Runs: 1
Failures: 0
Log Reference UI-Element Rollup
Info Clear
[info] Executing: |open | http://www.google.com | |
[info] Executing: |captureEntirePageScreenshot | d://google.png | |
[info] Executing: |break | | |


sel-ide-login-2-wp-com.html - Selenium IDE 1.10.0
File Edit Actions Options Help
Base URL http://wordpress.com/
Fast Slow
Test Case
sel-ide-login-2-wp-c...
Table Source
Command Target Value
open http://wordpress.com/
type name=log abcde@hujksaffds.com
type name=pwd testi
clickAndWait css=input[type="submi...
Command open
Target http://wordpress.com/
Find
Value
Runs: 1
Failures: 0
Log Reference UI-Element Rollup
Info Clear
[info] Executing: |type | name=log | abcde@hujksaffds.com |
[info] Executing: |type | name=pwd | testi |
[info] Executing: |clickAndWait | css=input[type="submit"] | |

DATE: / /
PAGE NO:

Experiment 8

Aim Study of any Test Management Tool (QA complete)

Theory

QA complete is a test management tool that can be used for both manual and automated testing. It is a tool with powerful test management capabilities that allows us to track all aspects of software quality in an efficient manner.

Benefits:

i) It can be integrated with any number of tools
ii) Customisable as per the tester's needs
iii) Requirements and tests can be traced to defect effectively
iv) Reports and dashboards are the key features QA complete.

Features:

1) Test Case Management: Simple test case structuring also allows for focused metrics and clear status report.
2) Test Environment Management: Various environments are linked to individual test cases for effective test coverage across different platforms, operating systems and devices.
3) Defect and Issue Management: Mainly tracks the resolution process of bugs for each release and automatically creates bugs when test cases fail.

Steps to setup and work on QA Complete

1. Login to QA complete tool.
2. Create a new release as E-work under the release tab and click add new item.

Sign.....

DATE
PAGE NO

3 Navigate to the test management tab → Test library → Add new folder.

4 Navigate to test management tab → test sets → create a folder create a new test set using the add new icon.

5 To run the test sets click test management tool → test sets → click the Run icon

6 Click the start run at the top right corner. Based on the working functionality, select the run status and click the end run options

7 If any of the steps fail in a test set, it prompts to create a defect

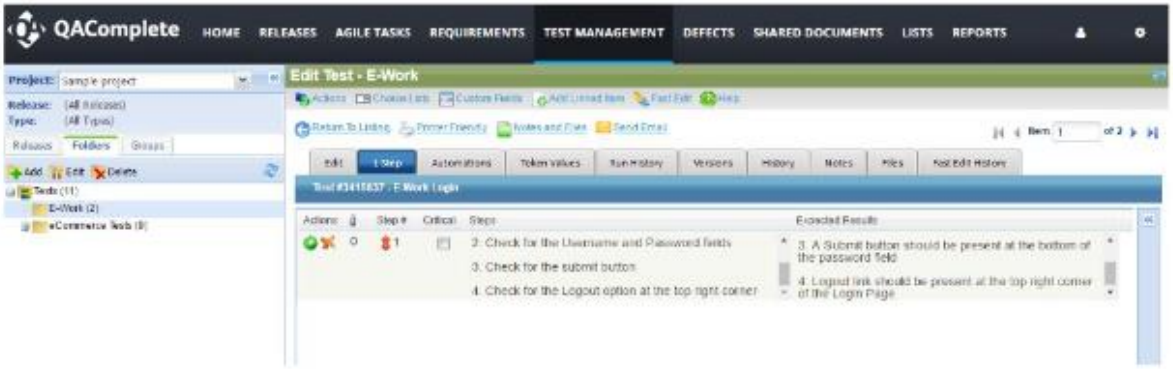
8 When the Yes option is selected a defect is created automatically

9 Navigate to the defects tab and view the automatically created bugs.

Result Hence the QA complete tool was successfully studied and explored.

Sign


QAComplete
HOME
RELEASES
AGILE TASKS
REQUIREMENTS
TEST MANAGEMENT
DEFECTS
SHARED DOCUMENTS
LISTS
REPORTS


QAComplete
HOME
RELEASES
AGILE TASKS
REQUIREMENTS
TEST MANAGEMENT
DEFECTS
SHARED DOCUMENTS
LISTS
REPORTS
Edit Test - E-Work


Edit Test Set - E-Work Test Set
Edit
2 Tests
Run History
History
Notes
Files
Choose Fields
Seq
Is Active?
Stop on Failure?
Id
Folder Name
Run By Host
Title
Status
1
3415837
E-Work
E-Work Login
New
2
3415984
E-Work
Login Page Field Validation - High Level Scenario
In Design