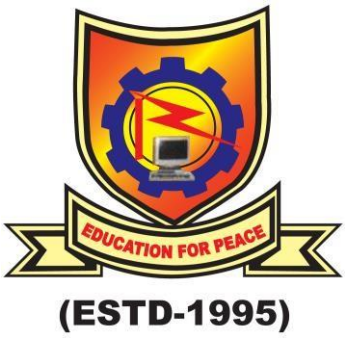
R G M College of Engineering and Technology (Autonomous)

Department of Computer Science & Engineering



SOFTWARE TESTING TOOLS

Lab Manual

IV B. Tech I SEM

RGM R-15 Regulations

VISION OF THE DEPARTMENT

- To empower students with cutting edge technologies in computer science and engineering
- To train the students as entrepreneurs in computer science and engineering to address the needs of the society
- To develop smart applications to disseminate information to rural people

MISSION OF THE DEPARTMENT

- To become the best computer science and engineering department in the region offering undergraduate, post graduate and research programs in collaboration with industry
- To incubate, apply and spread innovative ideas by collaborating with relevant industries and R & D labs through focused research groups.
- To provide exposure to the students in the latest tools and technologies to develop smart applications for the society

IV B.Tech. I-Sem (CSE) P C
3 2

(A1283156) SOFTWARE TESTING TOOLS LAB
(Common to CSE & IT)

OBJECTIVES:

Upon successful completion of this course students will be able to:

- Understand the basic concepts of software testing.
- Understand the various techniques and strategies of software testing and inspection and pointing out the importance of testing in achieving high-quality software.
- Perform effective and efficient structural testing of software.
- Integrate and test the various units and components of a software system.
- Perform effective and efficient functional testing of software.
- Select the appropriate tests to regression test your software after changes have been made.
- Plan, track and control the software testing effort.
- Understand the need of automated testing tools and various kinds of automated testing tools.

OUTCOMES:

- To understand the control structure of C program, Test cases, Test criteria, Test strategies and Testing Tools.
- To analyze the comparative study of Various Testing Techniques, and Tools.
- To design and conduct Manual Test Cases for a software testing project.
- To apply software testing tool to support test automation.

CO-PO MAPPING:

CO/PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	2	1		1		1	2			2	2	2	1	
CO2	3	3	3		2		2	2		1	3	1		2	1
CO3	3	1	3	3			1	3	3	3	3		1	1	
CO4	3		3	3	3		1	3	3	3	3			1	2

Lab Experiments:

1. Write programs in ‘C’ Language to demonstrate the working of the following constructs:
2. i) do...while
   ii) while….do
   iii) if…else
   iv) switch
   v) for
3. “A program written in ‘C’ language for Matrix Multiplication fails” Introspect the causes for its failure and write down the possible reasons for its failure.
4. Write manual test cases for Gmail application.
5. Write manual test cases for ATM application.
6. Write manual test cases for Banking application.
7. Study of Quick Test Professional(QTP):
8. Overview of QTP Components.
9. Record & Run Options.
10. Generating Basic Script.
11. Enhancement of Script.
12. Check Points.
13. Output Values.
14. Object Repository.
15. Writing Script manually.
16. Study of Rational Functional Tester(RFT).
17. Study of SELENIUM.

REFERENCES:

1. Software testing Tools – Dr.K.V.K.K.Prasad, Dreamtech.

Objective of the Laboratory: (Course Objectives)

Upon successful completion of this course students will be able to:

- Understand the basic concepts of software testing.
- Understand the various techniques and strategies of software testing and inspection and pointing out the importance of testing in achieving high-quality software.
- Perform effective and efficient structural testing of software.
- Integrate and test the various units and components of a software system.
- Perform effective and efficient functional testing of software.
- Select the appropriate tests to regression test your software after changes have been made.
- Plan, track and control the software testing effort.
- Understand the need of automated testing tools and various kinds of automated testing tools.

List of Experiments

Sl	Experiment (as stated in the syllabus)	Page Number
1	Write programs in 'C' Language to demonstrate the working of the following constructs: i) do...while ii) while….do iii) if…else iv) switch v) for	
2	"A program written in 'C' language for Matrix Multiplication fails" Introspect the causes for its failure and write down the possible reasons for its failure	
3	Write manual test cases for Gmail application.	
4	Write manual test cases for ATM application.	
5	Write manual test cases for Banking application.	
6	Study of Quick Test Professional(QTP)	
7	Overview of QTP Components.	
8	Record & Run Options.	
9	Generating Basic Script.	

10	Enhancement of Script.	
11	Check Points.	
12	Output Values	
13	Object Repository	
14	Writing Script Manually	
15	Study of Rational Functional Tester (RFT).	
16	Study of SELENIUM.	

Software Testing Tools Lab

Experiment No: 1a **Date:**

Aim:

Write a C program, which takes two integer operands and one operator from the user, performs the operation and then prints the result. (Consider the operations +, -, *, /, % and use switch statement).

Source Code:

```
#include<stdio.h>

#include<conio.h>


void main()

{

int a,b,res,ch;

clrscr();

printf("\t   **********************");

printf("\n\tMENU\n");

printf("\t**********************");

printf("\n\t(1)ADDITION");

printf("\n\t(2)SUBTRACTION");

printf("\n\t(3)MULTIPLICATION");

printf("\n\t(4)DIVISION");

printf("\n\t(5)REMAINDER");
```

```
printf("\n\t(0)EXIT");
printf("\n\t**********");
printf("\n\n\tEnter your choice:");
scanf("%d",&ch);

if(ch<=5 & ch>0)
{
printf("Enter two numbers:\n");
scanf("%d%d",&a,&b);
}

switch(ch)
{
 case1:
 res=a+b;
 printf("\n Addition:%d",res);


 case2:
 res=a-b;
 printf("\n Subtraction:%d",res);

 case3:
 res=a*b;
 printf("\n Multiplication:%d",res);


 case4
```

```
res=a/b;
printf("\n Division:%d",res);


case5:
res=a%b;
printf("\n Remainder:%d",res);
```

Input:

Case (i): **********

```
MENU
**********
(1)ADDITION
(2)SUBTRACTION
(3)MULTIPLICATION
(4)DIVISION
(5)REMAINDER
(0)EXIT
**********
Enter your choice: 3
Enter two numbers:
12
11
Multiplication: 132
```

Case (ii): **********

```
MENU
**********
(1)ADDITION
(2)SUBTRACTION
(3)MULTIPLICATION
```

(4)DIVISION

(5)REMAINDER

(0)EXIT

****

Enter your choice: 7

Expected Output:

Case (i): Multiplication: 132

Case (ii): Invalid Choice

Actual Output:

- Undefine symbol ‘case 4’ at line 49
- Statement missing ; at line 49
- Compound statement missing } at line 67
- Logical errors:
    - There is no space between keyword “case” and value.
    - There are no “break” statements after every case.

Deviation of actual result from the expected result:

- Errors instead of expected output.
- Since there is no space between keyword case and value, then default case is executed.
- Since there are no break statements all cases are executed.

Correction Code:

- Colon (:) is placed after “case 4” in line 49.
- The switch statement closing flower brackets are placed at line 67.
- Space is provided between keyword “case” and its value.
- “break;” keyword/ statement is added after all cases to terminate that case execution.

Results after Correction:

Case (i): Multiplication: 132

Case (ii): Invalid Choice

Experiment No: 1b **Date:**

Aim:

The total distance travelled by vehicle in 't' seconds is given by distance = ut+1/2at2 where 'u' and 'a' are the initial velocity (m/sec.) and acceleration (m/sec2). Write C program to find the distance travelled at regular intervals of time given the values of 'u' and 'a'. The program should provide the flexibility to the user to select his own time intervals and repeat the calculations for different values of 'u' and 'a'.

Source Code:

```
#include <stdio.h>

#include <math.h>


void main()

int tim_intrval, counter,time;

float accl, distance, velos;

clrscr();

printf("<===========PROGRAM FOR CALC TOTAL DISTANCE TRAVELED BY A VEHICAL==========>");

printf("\n\n\n\t\t\tNO OF TIME INTERVALS : ");

scanf("%f",&tim_intrval);


for(counter = 1; counter <= tim_intrval; counter+)

{

    printf("\n\t\t\tAT T%d TIME(sec) : ",counter);

    scanf("%d",&time);

    printf("\t\t\tVELOCITY AT %d sec (m/sec) : ",time);

    scanf("%f",&velos);

    printf("\t\t\tACCLERATION AT %d sec (m/sec^2): ",time);

    scanf("%f",&accl);

    distance += (velos*time + (accl*pow(time,2)/2);

}
```

```
printf("\n\n\n\tTOTAL DISTANCE TRAVELLED BY VEHICLE IN %d INTERVALS OF
TIME : %f",tim_intrval,&distance);

getch();

}
```

Input:

Case (i):

```
<===========PROGRAM FOR CALC TOTAL DISTANCE TRAVELED BY A
VEHICLE===========>

NO OF TIME INTERVALS :1

AT T1 TIME(sec) : 10

VELOCITY AT 10 sec (m/sec) : 5

ACCLERATION AT 10 sec (m/sec^2): 10
```

Case (ii):

```
<===========PROGRAM FOR CALC TOTAL DISTANCE TRAVELED BY A
VEHICLE===========>

NO OF TIME INTERVALS :2

AT T1 TIME(sec) : 10

VELOCITY AT 10 sec (m/sec) : 4

ACCLERATION AT 10 sec (m/sec^2): 5


AT T2 TIME(sec) : 5

VELOCITY AT 5 sec (m/sec) : 10

ACCLERATION AT 5 sec (m/sec^2): 5
```

Expected Output:

Case (i): TOTAL DISTANCE TRAVELLED BY VEHICLE IN 1 INTERVALS OF TIME: 550.000000

Case (ii): TOTAL DISTANCE TRAVELLED BY VEHICLE IN 2 INTERVALS OF TIME: 402.500000

Actual Output:

- Unexpected end of file in comment started on line 1
- Declaration syntax error in line 12,19
-) expected in line 15,16,17
- Declaration terminated incorrectly at line 19
- Unexpected } at line 32
- Logical Errors:
    - Usage of incorrect format specifier i.e; %f for int datatype in scanf().
    - Usage of "&" to print the variable (before arguments) in printf().

Deviation of actual result from the expected result:

- Errors instead of expected output.
- Since %f is used for variable "tim_interval", it doesn't satisfy the loop condition & loop doesn't executed, because its datatype is "int"
- As we use "&" in printf() to print value of "distance" variable, it results incorrect output.

Correction Code:

- The comment lines are closed at required line.
- The increment operator is placed according to the for loop.
- All the opening paranthesis are matched with a proper closing paranthesis.
- %d format specifier is placed for int datatype variable "tim_interval" in scanf()
- The "&" symbol is removed before argument in printf()

Results after Correction:

Case (i): TOTAL DISTANCE TRAVELLED BY VEHICLE IN 1 INTERVALS OF TIME: 550.000000

Case (ii): TOTAL DISTANCE TRAVELLED BY VEHICLE IN 2 INTERVALS OF TIME: 402.500000

Experiment No: 1c **Date:**

Aim:

Write a C program to calculate the following Sum: **Sum=1-x2/2! +x4/4!-x6/6!+x8/8!-x10/10!**

Source Code:

```
#include <stdio.h>

void main(
{
int counter,f_coun;
float sum=0,x,power,fact
clrscr();

printf("<----------------------PROGRAM FOR SUM OF EQ. SERIES---------------------->");
printf("\n\n\tEQUATION SERIES : 1- X^2/2! + X^4/4! - X^6/6! + X^8/8! - X^10/10!");

printf("\n\n\n\tENTER VALUE OF X : ");
scanf("%d",&x);

for(counter=0;power=0; power<=10; counter++;power=power+2)
{
fact=1;
//CALC FACTORIAL OF POWER VALUE
for(f_coun=power; f_coun>=1; f_coun--)
      fact *= f_coun;
//EQ. FOR SUM SERIES
sum=sum+(pow(-1,counter)*(pow(x,power)/fact);
}
```

```
printf("SUM : %f",sum);
getch();

}
```

Input:

Case (i):

<----------PROGRAM FOR SUM OF EQ. SERIES---------->

EQUATION SERIES: 1- X^2/2! + X^4/4! - X^6/6! + X^8/8! - X^10/10!

ENTER VALUE OF X: 2

Case (ii):

<----------PROGRAM FOR SUM OF EQ. SERIES---------->

EQUATION SERIES: 1- X^2/2! + X^4/4! - X^6/6! + X^8/8! - X^10/10!

ENTER VALUE OF X: 4

Expected Output:

Case (i): SUM: -0.416155

Case (ii): SUM: -0.685785

Actual Output:

- Expected at line 8.
- Declaration syntax error at line 11.
- Possibly incorrect assignment at line 19.
- For statement missing) at line 19.
- Statement missing; at line 19.
- "Sum" is assignment a value is never used at line 32.
- Logical Errors:
    - Undefined header file "math.h"
    - Usage of incorrect format specifier i.e; %d for float datatype in scanf()

Deviation of actual result from the expected result:

- Errors instead of expected output.
- Incorrect output due to usage of "pow()" without including "math.h" header file i.e; pow: OVERFLOW error
- Since %d is used for float variable, it results in incorrect output.

Correction Code:

- Close the main() method according to the proper syntax.
- End the statement with semi-colon at line 10.
- Separate the initialization and increment according to proper syntax in for loop.
- Match all opening parenthesis with closing parenthesis at line 25.
- Include the "<math.h>" header file in the program.
- Place %f format specifier instead of %d in scanf() for variable "X".

Results after Correction:

Case (i): SUM: -0.416155

Case (ii): SUM: -0.685785

Experiment No: 1d **Date:**

Aim:

A Fibonacci Sequence is defined as follows: the first and second terms in the sequence are 0 and 1. Subsequent terms are found by adding the preceding two terms in the sequence. Write a C program to generate the first n terms of the sequence.

Source Code:

```
#include <stdio.h>


void main()

{

int num1=0, num2=1,no,counter,fab;

clrscr(;


printf("<===========PROGRAM TO FIND THE FIBONACCI SERIES UP TO N NO. IN SERIES==========>");

printf("\n\n\n\t\tENTER LENGTH OF SERIES (N) : ");

scanf("%f",&no);


printf("\n\n\t\t\t<----FIBONACCI SERIES---->");

printf("\n\n\t\t%d  %d",&num1,num2);


//LOOP WILL RUN FOR 2 TIME LESS IN SERIES AS THESE WAS PRINTED IN ADVANCE

for(counter = 1; counter <= no-2; counter+)

{

fab=num1 + num2;

printf("  %d",&fab);

num1=num2;

num2=fab;

}
```

getch();

}

Input:

Case (i):

<==========PROGRAM TO FIND THE FIBONACCI SERIES UP TO N NO. IN SERIES=========>

ENTER LENGTH OF SERIES (N): 5

Case (ii):

<==========PROGRAM TO FIND THE FIBONACCI SERIES UP TO N NO. IN SERIES=========>

ENTER LENGTH OF SERIES (N): 10

Expected Output:

Case (i):

<----FIBONACCI SERIES---->

0 1 1 2 3

Case (ii):

<----FIBONACCI SERIES---->

0 1 1 2 3 5 8 13 21 34

Actual Output:

- Expression syntax at line 13,23.
- Logical Errors:
    - Incorrect format specifier %f is used for “int” variable “no”.
    - Usage of “2” to print the variable “num1” in printf().
    - Usage of “&” to print the variable “fab” in printf()

Deviation of actual result from the expected result:

- Errors instead of expected output.
- Usage of “&” in used printf() displays incorrect output.
- Since %f is used for int “no”, scanf() is performed incorrectly.

Correction Code:

- The “clrscr()” is edited as per the syntax.
- The increment expression is edited as per the syntax of for loop.
- Place %d format specifier instead of %f in scanf() for variable “no”.

Results after Correction:

Case (i):

<----FIBONACCI SERIES---->

0 1 1 2 3

Case (ii):

<----FIBONACCI SERIES---->

0 1 1 2 3 5 8 13 21 34

Experiment No: 1e **Date:**

Aim:

A Fibonacci Sequence is defined as follows: the first and second terms in the sequence are 0 and 1. Subsequent terms are found by adding the preceding two terms in the sequence. Write a C program to generate the first n terms of the sequence.

Source Code:

```
#include <stdio.h>


void main()

{

int num1=0, num2=1,no,counter,fab;

clrscr(;


printf("<===========PROGRAM TO FIND THE FIBONACCI SERIES UP TO N NO. IN SERIES==========>");

printf("\n\n\n\t\tENTER LENGTH OF SERIES (N) : ");

scanf("%f",&no);


printf("\n\n\t\t\t<----FIBONACCI SERIES---->");

printf("\n\n\t\t%d  %d",&num1,num2);


//LOOP WILL RUN FOR 2 TIME LESS IN SERIES AS THESE WAS PRINTED IN ADVANCE

for(counter = 1; counter <= no-2; counter+)

{

fab=num1 + num2;

printf("  %d",&fab);

num1=num2;

num2=fab;

}
```

getch();

}

Input:

Case (i):

<==========PROGRAM TO FIND THE FIBONACCI SERIES UP TO N NO. IN SERIES=========>

ENTER LENGTH OF SERIES (N): 5

Case (ii):

<==========PROGRAM TO FIND THE FIBONACCI SERIES UP TO N NO. IN SERIES=========>

ENTER LENGTH OF SERIES (N): 10

Expected Output:

Case (i):

<----FIBONACCI SERIES---->

0 1 1 2 3

Case (ii):

<----FIBONACCI SERIES---->

0 1 1 2 3 5 8 13 21 34

Actual Output:

- Expression syntax at line 13,23.
- Logical Errors:
    - Incorrect format specifier %f is used for “int” variable “no”.
    - Usage of “2” to print the variable “num1” in printf().
    - Usage of “&” to print the variable “fab” in printf()

Deviation of actual result from the expected result:

- Errors instead of expected output.
- Usage of “&” in used printf() displays incorrect output.
- Since %f is used for int “no”, scanf() is performed incorrectly.

Correction Code:

- The “clrscr()” is edited as per the syntax.
- The increment expression is edited as per the syntax of for loop.
- Place %d format specifier instead of %f in scanf() for variable “no”.

Results after Correction:

Case (i):

<----FIBONACCI SERIES---->

0 1 1 2 3

Case (ii):

<----FIBONACCI SERIES---->

0 1 1 2 3 5 8 13 21 34

Experiment No: 1f **Date:**

Aim:

Write C program that use recursive function to find the factorial of a given integer.

Source Code:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n;
        clrscr();
        printf("Enter the number :");
        scanf("%d",&n);
        printf("Factorial of number %d",factorial(n));
        getch();
        int fact(int n)
        int f;
        if((n=0)|(n=1)) // check the condition for the n value
        return(n);
        else
        f=n*fact(n-1); //calculate the factorial of n
        return(f);
}
```

Input:

Case (i): Enter the number: 5

Case (ii): Enter the number: 8

Expected Output:

Case (i): Factorial of number 120

Case (ii): Factorial of number 40320

Actual Output:

- Declaration is not allowed here at line 13
- Declaration syntax error at line 14
- Void function may not return a value at line 16
- Undefined symbol 'f' at line 18
- Void function may not return a value at line 19
- Logical errors:
    - Undefined method name is used in the function call
    - Improper condition evaluation in the if statement i. e; = instead of "=="
    - Small datatype variable 'int' is used to store the result of long datatype.
    - Type mismatch in redeclaration of function "fact".

Deviation of actual result from the expected result:

- Errors instead of expected output.
- A runtime error is generated due to the calling of undefined method name factorial()
- Improper evaluation of if condition which results in incorrect output.
- Since the type of output variable of function fact() is 'int', it can't display results exceeding range of 'int'.
- Function declaration of fact() is not recognized by main().

Correction Code:

- main() is closed properly at required position.
- Function definition of fact() is to be edited properly according to the syntax.
- The function call is to be made with correct declared function name.
- Place '==' incase of '=' in if condition.
- Specify the return type of function fact() from int to long int and place the format specifier as "%ld" in printf() to print returned value.
- Place the function declaration of the fact() at the top the main().

Results after Correction:

Case (i): Factorial of number 120

Case (ii): Factorial of number 40320

Experiment No: 1g **Date:**

Aim:

Write C program that use recursive function to find the factorial of a given integer.

Source Code:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n;
        clrscr();
        printf("Enter the number :");
        scanf("%d",&n);
        printf("Factorial of number %d",factorial(n));
        getch();
        int fact(int n)
        int f;
        if((n=0)|(n=1)) // check the condition for the n value
        return(n);
        else
        f=n*fact(n-1); //calculate the factorial of n
        return(f);
}
```

Input:

Case (i): Enter the number: 5

Case (ii): Enter the number: 8

Expected Output:

Case (i): Factorial of number 120

Case (ii): Factorial of number 40320

Actual Output:

- Declaration is not allowed here at line 13
- Declaration syntax error at line 14
- Void function may not return a value at line 16
- Undefined symbol 'f' at line 18
- Void function may not return a value at line 19
- Logical errors:
    - Undefined method name is used in the function call
    - Improper condition evaluation in the if statement i. e; = instead of "=="
    - Small datatype variable 'int' is used to store the result of long datatype.
    - Type mismatch in redeclaration of function "fact".

Deviation of actual result from the expected result:

- Errors instead of expected output.
- A runtime error is generated due to the calling of undefined method name factorial()
- Improper evaluation of if condition which results in incorrect output.
- Since the type of output variable of function fact() is 'int', it can't display results exceeding range of 'int'.
- Function declaration of fact() is not recognized by main().

Correction Code:

- main() is closed properly at required position.
- Function definition of fact() is to be edited properly according to the syntax.
- The function call is to be made with correct declared function name.
- Place '==' incase of '=' in if condition.
- Specify the return type of function fact() from int to long int and place the format specifier as "%ld" in printf() to print returned value.
- Place the function declaration of the fact() at the top the main().

Results after Correction:

Case (i): Factorial of number 120

Case (ii): Factorial of number 40320

Experiment No: 1h **Date:**

Aim:

Write a C program to find both the largest and smallest number in a list of integers.

Source Code:

```
#include<stdio.h>
void main()
{
int a[10],i,n,min,max
clrscr();
printf("enter the array size:");
scanf("%d",n);
printf("Enter the elements of array");
for(i=0;i<n;i++); // read the elements of an array
scanf("%d",&a[i]);
min=a[0];
max=a[0];
for(i=0;i<n;i++)// read the elements of an array
{
if(a[i]<min)// check the condition for minimum value
min=a[i];
if(a[i]>max)//check the condition for maximum value
max=a[i];
}
printf("maximum value is:%d\n",max);
printf("minimum value is:%d\n",min);
getch();
```

}

Input:

Case (i): Enter the array size: 5

Enter the elements of array

-999 -998 -4 -3 -1

Case (ii): Enter the number: 5

Enter the elements of array

2 10 -5 20 30

Expected Output:

Case (i):

Minimum value is: -1

Maximum value is: -999

Case (ii):

Minimum value is: -5

Maximum value is: 30

Actual Output:

- Declaration syntax error at line 7
- Logical errors:
    - Improper syntax used for reading a variable in scanf() without using '&'
    - Improper declaration of ';' at end of the for loop for reading all the array elements.

Deviation of actual result from the expected result:

- Errors instead of expected output.
- The value is not properly stored in the variable 'n'.
- The loop is not repeated 'n' no of times & terminates by reading one value.

Correction Code:

- Place the ';' at end of variable declaration at lines.
- Place the '&' before the variable 'n' in scanf().
- Remove ';' at the end of the for loop for reading all array elements.

Results after Correction:

Case (i):

Minimum value is: -1

Maximum value is: -999

Case (ii):

Minimum value is: -5

Maximum value is: 30

Experiment No: 1J **Date:**

Aim:

Write a C program to generate all the prime numbers between 1 and n, where n is a value supplied by the user.

Source Code:

```
#include <stdio.h>


void main()

{

int no,counter,counter,check;

clrscr();

printf("<----------------------PRIME NO. SERIES---------------------->");

printf("\n\n\n\t\t\tINPUT THE VALUE OF N: ");

scanf("%d",&no);

printf("\n\nTHE PRIME NO. SERIES B/W 1 TO %d : \n\n",no);


for(counter = 1; counter <= no; counter++)

{

 check == 0;

 //THIS LOOP WILL CHECK A NO TO BE PRIME NO. OR NOT.


 for(counter1 = counter-1; counter1 > 1 ; counter1--)

   if(counter%counter1 = 0)

   {

    check++;        // INCREMENT CHECK IF NO. IS NOT A PRIME NO.

    break;

   }
```

```
  if(check = 0)
  printf("%d\t",counter);
}
getch();
}
```

Input:

Case (i):

<----------PRIME NO. SERIES---------->

INPUT THE VALUE OF N:10

Case (ii):

<----------PRIME NO. SERIES---------->

INPUT THE VALUE OF N:15

Expected Output:

Case (i):

THE PRIME NO. SERIES B/W 1 TO 10:

1 2 3 5 7

Case (ii):

THE PRIME NO. SERIES B/W 1 TO 15:

1 2 3 5 7 11 13

Actual Output:

- Unexpected end of file in comment started on line1.
- Multiple declaration for ‘counter’ at line 7.
- Code has no effect at line 16.
- Undefined symbol ‘counter1’ at line 19.
- Possibly incorrect assignment at line 25.

Deviation of actual result from the expected result:

- Errors instead of expected output.

Correction Code:

- Comments lines are closed at required line.
- Declaration of multiple variables with same name is not allowed, so re-name any one of the variable.

- By using assignment operator, we can initialize value to variable but here comparison operator is used which has different effect on the code.
- Variable ‘counter1’ is used in program without declaration. So, declare the ‘counter1’ variable in program.
- ‘==’ is used in if statement instead of ‘=’.

Results after Correction:

Case (i):

THE PRIME NO. SERIES B/W 1 TO 10:

1 2 3 5 7

Case (ii):

THE PRIME NO. SERIES B/W 1 TO 15:

1 2 3 5 7 11 13

Experiment No: 1K **Date:**

Aim:

Write a C program to find the roots of a quadratic equation.

Source Code:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>

void main()
{
int a,b,c,root1,root2;
clrscr();
printf("\n Enter values of a,b,c for finding roots of a quadratic eq:\n");
scanf("%f%f%f",&a,&b,&c);

/*checking condition*/
if(b*b>4*a*c)
{
 root1=-b+sqrt(b*b-4*a*c)/2*a;
 root2=-b-sqrt(b*b-4*a*c)/2*a;
 printf("\n*****ROOTS ARE*****\n");
 printf("\n root1=%f\n root2=%f",&root1,&root2);

else
 printf("\n Imaginary Roots.");
 getch;
```

Input:

Case (i):
Enter values of a,b,c for finding roots of a quadratic eq:

1

-5

6

Case (ii):

Enter values of a,b,c for finding roots of a quadratic eq:

1

-10

24

Expected Output:

Case (i):

*****ROOTS ARE*****

Root1=3

Root2=2

Case (ii):

*****ROOTS ARE*****

Root1=6

Root2=4

Actual Output:

- Misplaced else at line 22.
- Code has no effect at line 24
- Compound statement missing } at line 24,27
- Logical error:
    - Incorrect format specifier %f is used for 'int' variables a,b,c.
    - Usage of '&' to print the variables i.e root1, root2.
    - Incorrect format specifier %f is used for printing the variables root1, root2.
    - Incorrect condition in if statement for evaluating the roots of an equation.
    - Placing the paranthesis at required position is not done proper to calculate the roots.

Deviation of actual result from the expected result:

- Errors instead of expected output.

- Due to the incorrect format specifier %f the'int' values are not stored properly.
- Due to the incorrect usage of '&' in printf(), it displays address of variable.
- Due to incorrect format specifier %f is used for printing the integer variables, instead of values zeros are printed.
- The condition is not evaluated due to the incorrect condition.
- Evaluation of an expression can be done improperly due to improper placing of paranthesis.

Correction Code:

- Opening and closing braces must be matched for if condition and main()
- Put the paranthesis properly for getch()
- Replace the %f with the %d in both printf() and scanf().
- Remove '&' before the variable in printf().
- Modify the condition in if statement as per the logic required.
- Place the parenthesis at required position.

Results after Correction:

Case (i):

*****ROOTS ARE*****

Root1=3

Root2=2

Case (ii):

*****ROOTS ARE*****

Root1=6

Root2=4

Experiment No: 1L **Date:**

Aim:

Write a C program to find the sum of individual digits of a positive integer.

Source Code:

```
#include<stdio.h>
#include<conio.h>


void main()
  int num k=1, sum=0;
  clrscr();
  printf("Enter the number whose digits are to be added:");
  scanf("%d",num);
while(num!=0)
{
  k=num%10;
  sum=sum+k;
  k=num%10;
  num=k;
}
printf("Sum of the digits:%d",&sum);
getch();
}
```

Input:

Case (i):

Enter the number whose digits are to be added:

1234

Case (ii):

Enter the number whose digits are to be added:

786

Expected Output:

Case (i):

Sum of the digits: 10

Case (ii):

Sum of the digits: 21

Actual Output:

- Declaration syntax error at line 7
- Type mismatch in redeclaration of 'clrscr' at line 8
-) expected at line 9,10
- Declaration Terminated incorrectly at line 11.
- Unexpected } at line 20.
- Logical error:
    - Specify '&' in scanf() for reading variable.
    - Usage of '&' in printf() to print the 'sum' variable.
    - Instead of '%' we need to specify '/' in while loop.

Deviation of actual result from the expected result:

- Errors instead of expected output.
- As '&' is not used in scanf(), the value of variable is not stored properly.
- Address value is printed instead of actual value due to the usage of '&' in printf() before variable name.
- The logic is not evaluated properly due to incorrect operators usage.

Correction Code:

- Opening and closing braces of main() should be properly placed.
- Declaration of multiple variables in the same line should be separated by using the ','
- Place the '&' before the variable name in the scanf() for reading
- Remove the '&' before the printing variable name in printf()
- In order to evaluate the logic correctly replace the % with '/' in while loop.

Results after Correction:

Case (i):

Sum of the digits: 10

Case (ii):

Sum of the digits: 21

Software Testing

Software testing methods:

In general, organizations follow 2 types of methods for validating an application.

1. Manual Testing
2. Automation Testing

Manual Testing:

Manual Testing is a process in which without using any automation test tool, test engineers will verify the actual behavior of the application while performing operations on the application.

Advantages:

1. It is easy and simple to perform.

Disadvantages:

1. Time consuming
2. Lack of reliability
3. Lack of consistency

Automation Testing:

It is a process of automating human activities in order to validate the application.

Advantages:

1. Fast in execution
2. More reliable
3. More consistent
4. Automation scripts are reusable.

Disadvantages:

1. Automation tools are expensive
2. Skilled automation test engineers are required.

3. Tools may not support different environments

3,4 &5 Example for manual Test cases template for Gmail ,ATM & Bank

Manual Testing (Writing Test cases) for Employee Search

TCID\name	Test case description	Priority	Step name	Step description	Expected results
TC01-CIS-empsearch-Invoke app	This tc to verify the application and to check availability of components	High	Step1	Enter valid URL	Employee search page should open
TC02-CIS-emp search wit empname	This tc to verify the search results using emp name	High	Step1	Enter valid emp name and click on search	Corresponding employee record should display
			Step 2	Enter invalid emp name and click on search	Corresponding employee record should not display
TC03-CIS-emp search –with dept	This tc to verify the search results using emp dept	High	Step1	Enter valid emp dept and click on search	Corresponding employee department should display
			Step2	Enter invalid emp dept and click on search	Error message should displayed or pop –up window displayed
TC04-CIS-emp search –with DOJ	This tc to verify the search results using emp DOJ	High	Step1	Enter valid emp date of joining	All the employee record should displayed with corresponding dates
			Step2	Enter invalid emp DOJ and click on search	Error message should displayed or pop –up window displayed
TC05-emp	This tc to	high	Step1	Enter data in	Error message

search –with multiple data	verify the search results using multiple data			many combinations emp name, emp dept, emp DOJ and click on search	should displayed or pop –up window displayed
TC06-CIS-emp search –without data	This tc to verify the search results without data	High	Step1	Click on serch button without entering the data	Error message should displayed or pop –up window displayed
TC07-CIS-emp search –with clear button functionality	This TC to verify clear button functionality	Mediu m	Step1	Click on clear button after getting search results	Data should be cleared in search page

6. QTP (Quick Test Professional)

1. It is a product of HP.

2. It is a functionality and regression testing tool.

3. It is compatible with windows OS only.

4. It supports client/server applications and web-based applications.

QTP supports VB script and JavaScript for automation scripting.

Basic working principle of QTP is record and playback.

Record:

By default QTP is able to convert our operations which we perform on AUT (Application under Test) into automation scripts.

Playback:

During script runtime QTP will perform same operation on AUT with respect to script.

7. Over View of Components of QTP

Components in QTP Main screen:

1. Tool Bar:

It contains menu options and icons to perform operations on QTP

2. Test Pane:

It is like an Editor screen, where we can generate automation script and we can perform some editing operations

Automation script we can generate using recording modes in QTP or by writing script manually

In Test Pane there are 2 types of views:

a. Expert View:
In this view by default script generates in VB Script

b. Keyword View:
In this view script generates in simple understandable language in terms of "Item", "Operation", "Value" and "Documentation"

Note: in general we prefer "Expert View" where it is easy to write the script manually

Keyword view→ easy to understand the script without Vb script knowledge

3. Active Screen:

During recording by default QTP captures snapshot of application for each operation and those will be maintained in Active screen component

Advantages of active screen:

a. easy to understand script

b. we can perform some editing operations in the script like inserting checkpoints, output values, new steps…etc
c. we can view/add test object properties into Object Repository

Disadvantage:

Active screen files will occupy more memory space

4. Data table:

In QTP we have built-in data table where we can import/store required test data and from that we can parameterize test script during runtime

*there are 2 types of sheets in Data table

a. Global sheet

b. Action/Local sheet

Note: in QTP within the Test we can create maximum 255 Actions, for each action QTP provides individual action sheets in Data table

a. Global sheet:
By default script will execute multiple times based on number of rows filled with test data in Global sheet
using test data from Global sheet we can parameterize any action script

b. Action/local Sheet:
Irrespective of number of rows filled with test data in Action/local sheet script will execute only one time

Using test data from Action sheet we can parameterize that particular action script only

5. Test Flow:

In this component we can view the sequence of actions execution flow in a test and also we can re-order those actions execution flow by performing Drag and Drop option

In general Actions will execute, in which sequence we created those actions in a test

6. Debug Viewer:

During execution break time to view the intermediate values of variables and to update those values we use Debug viewer

In Debug viewer we have 3 sections

a. Watch → to view specific variable value
b. Variables → to view all the variable values
c. Command→ to update the value in a variable

7. Information pane: (Ctrl+F7)

This component will provide syntax error information in the script

8. Missing Resources:

In general for a test we associate different resource files like shared repositories, recovery scenarios, environment variables, test data, library functions…etc

For a opened test if any associated resource file is not available that information we can find in "Missing Resources" component

8. Record and Run Options in QTP:

The steps in testing process are:

1. Plan the test:

Identify automation test scenario which we need to test and analyze.

Eg: Write test scenario for performing operations on calculator application.

Step 1: Activate Calculator.

Step 2: Click on "C".

Step 3: Click on "5"

Step 4: Click on "*"

Step 5: Click on "6"

Step 6: Click on "="

Test result = 30

2. Prepare the tool for recording:

By default, QTP is able to record multiple application operations in to automation script.

There are 2 types of options for recording:

a) Record and run on any open window based application. When we select this option, QTP is able to record multiple application operations into automation scripts.

Navigation steps:

Step 1: Go to "Automation" menu

Step 2: Select "Record" option

Step 3: Select windows application tab in records and run settings window.

Step 4: Select first option of recording.

b) Record and run only on.

Navigation steps:

Step 1: Go to “Automation” menu

Step 2: Select “Record” option

Step 3: Go to windows application tab in records and run settings window.

Step 4: Select 2nd option of recording.

Step 5: Click on “+” button

Step 6: Click on “Browse” button

Step 7: Browse for application

Step 8: Select “launch application” option, if required.

Step 9: Click on “Ok”

Step 10: Click on “Ok”.

3. Generating basic script:

Based on automation scenario we generate basic script to perform operation on AUT.

Experiment No: 4a **Date:**

Aim:

Create a basic script to perform operations on calculator application using first option of recording.

Test Scenario:

Step 1: Activate Calculator.

Step 2: Click on “C”.

Step 3: Click on “3”

Step 4: Click on “*”

Step 5: Click on “4”

Step 6: Click on “=”

Test result = 12

Navigation:

Step 1: Open QTP

Step 2: Go to “Automation” menu. Select “Record” option.

Step 3: Select windows application tab in Records and run settings window.

Step 4: Select first option of Recording.

Generated Basic script:

1. Window(“Calculator”).Activate
2. Window(“Calculator”).Move 450,300
3. Window(“Calculator”).winButton(“C”).click
4. Window(“Calculator”).winButton(“3”).click
5. Window(“Calculator”).winButton(“*”).click
6. Window(“Calculator”).winButton(“4”).click
7. Window(“Calculator”).winButton(“=”).click
8. Window(“Calculator”).Minimize

Test Results summary:

Iteration #	Results
1	Done

Status	Times
Passed	1
Failed	0
Warnings	0

Experiment No: 4b **Date:**

Aim:

Create a basic script to perform operations on calculator application using second option of recording.

Test Scenario:

Step 1: Activate Calculator.

Step 2: Click on "C".

Step 3: Click on "3"

Step 4: Click on "*"

Step 5: Click on "4"

Step 6: Click on "="

Test result = 12

Navigation:

Step 1: Go to "Automation" menu

Step 2: Select "Record" option

Step 3: Go to windows application tab in records and run settings window.

Step 4: Select 2nd option of recording.

Step 5: Click on "+" button

Step 6: Click on "Browse" button

Step 7: Browse for application

Step 8: Select "launch application" option, if required.

Step 9: Click on "Ok"

Step 10: Click on "Ok".

Generated Basic script:

1. Window(“Calculator”).Activate
2. Window(“Calculator”).Move 450,300
3. Window(“Calculator”).winButton(“C”).click
4. Window(“Calculator”).winButton(“3”).click
5. Window(“Calculator”).winButton(“*”).click
6. Window(“Calculator”).winButton(“4”).click
7. Window(“Calculator”).winButton(“=”).click
8. Window(“Calculator”).Close

Test Results summary:

Iteration #	Results
1	Done

Status	Times
Passed	1
Failed	0
Warnings	0

Experiment No: 5a **Date:**

Aim: Create a basic script to open 5th order number record in flight reservation application.

Test Scenario:

Step 1: Activate 'Flight Reservation' application.

Step 2: Go to File → open order

Step 3: Select order number in open order dialog

Step 4: Enter order number 5

Step 5: Click on ok button.

Navigation:

Step 1: Open QTP

Step 2: Go to "Automation" menu. Select "Record" option.

Step 3: Select windows application tab in Records and run settings window.

Step 4: Select first window.

Generated Basic script:

1. Window("Flight Reservation").Activate
2. Window("Flight Reservation").winMenu("Menu").select "File"; open order… 450,300
3. Window("Flight Reservation").Dialog("Open Order").winCheckBox("Order No.").set "ON"
4. Window("Flight Reservation").Dialog("Open Order").click 68.179
5. Window("Flight Reservation").Dialog("Open Order").winEdit("Edit").set "5"
6. Window("Flight Reservation").Dialog("Open Order").winButton("Ok").click
7. Window("Flight Reservation").winButton("Update order").click

8. Window("Flight Reservation").Minimize

Test Results summary:

Iteration #	Results
1	Done

Status	Times
Passed	1
Failed	0
Warnings	0

Experiment No: 5b **Date:**

Aim: Create a basic script for booking a flight in flight reservation application.

Test Scenario:

Step 1: Activate 'Flight Reservation' application.

Step 2: Go to File → New order

Step 3: Enter valid date(mm/dd/yy)

Step 4: select source in fly from list box

Step 5: select destination in fly to list box

Step 6: Click on flights button

Step 7: Select flight timings in flights table

Step 8: Enter valid name

Step 9: Select class of Journey

Step 10: Click on insert order

Navigation:

Step 1: Open QTP

Step 2: Go to "Automation" menu. Select "Record" option.

Step 3: Select windows application tab in Records and run settings window.

Step 4: Select first option.

9.Generated Basic script:

1. Window("Flight Reservation").Activate

2. Window("Flight Reservation").winMenu("Menu").select "File.New order"

3. Window("Flight Reservation").winObject("Date of flight").type "121420"

4. Window("Flight Reservation").winComboBox("Fly from").select "LosAngeles"

5. Window("Flight Reservation").winButton("Fly To").select "Paris"

6. Window("Flight Reservation").winButton("Flight").click

7. Window("Flight Reservation").Dialog("Flight Table").winList("From").select "19078 LON 10:24 AM LAX 05:31 PM LH $153.40"

8. Window(“Flight Reservation”).Dialog(“Flights Table”).winButton(“Ok”).click

9. Window(“Flight Reservation”).winEdit(“Name”).set “LS”

10. Window(“Flight Reservation”).winRadioButton(“Business”).set

11. Window(“Flight Reservation”).winButton(“Insert Order”).click

12. Window(“Flight Reservation”).Close

Test Results summary:

Iteration #	Results
1	Done

Status	Times
Passed	1
Failed	0
Warnings	0

10. **Enhancement of the Basic Script in QTP**

In QTP script should be enhanced with the help of Check points and output values i.e for whatever script generated for recording after that have to insert check point in order to validate the question on given requirement so by putting check points or output values enhanced the script written.

11. Check points

. There are 10 checkpoints in QTP

1. Standard checkpoint
2. Text checkpoint
3. Text Area checkpoint
4. Bitmap checkpoint
5. Database checkpoint
6. XML checkpoint
7. Accessibility checkpoint
8. Image checkpoint
9. Table checkpoint
10. Page checkpoint

Basic working principle of checkpoint:-

1. While creation time of checkpoint the test engineers should provide their expected value to checkpoint
2. During runtime the checkpoint captures the actual value from AUT

Standard Checkpoint:

Using this standard checkpoint we can verify the standard property values an application object without expected values.

Example properties: Enabled, items count, content, item selection et.,

Create basic script to verify the selected item in fly from list box with expected calue as "paris" or 5thorder number record

Test Scenario:

Step1 : create basic script for opening 5th order number record

Step2: create standard checkpoint to verify the selected item in flyfrom listbox as paris

Navigation:

1. Select position in the script
2. Enter into recording mode
3. Goto insert menu
4. Select checkpoint- standard checkpoint
5. Move hand icon on to the corresponding element and click on it
6. Click on ok after verification in object selection property window
7. Enter expected value in the constant option
8. Click on ok

Generated Script:

Window(“Flight Reservation”).Activate

Window(“Flight Reservation”).WinMenu(“Menu”).select “File Open Order”

Window(“Flight Reservation”).Dialog(“OpenOrder”).WinCheckBox(“Order No”)

Window(“Flight Reservation”).Dialog(“OpenOrder”).WinEdit(“Edit”).sert ‘5’

Window(“Flight Reservation”).Dialog(“OpenOrder”).WinButton(“on”).click

Window(“Flight Reservation”).WinComboBox(“Fly From”).checkCheckPoint(“Fly From”)

Table Results:

Test Result Summary

Iterations	Summary
1	Failed

Status	Times
Passed	0
Failed	1
Warnings	0

Details

Selection	Paris
	Denver

Parameterizing Checkpoints:

Sometimes we can pass multiple expected values to a checkpoint during runtime this process is called parameterizing checkpoints.

Create a basic script to verify update order button enable property with following expected values after booking a flight

Expected values : True False True False

Test Scenario: -

Step1: Enter expected values into the global sheet of database

Step2: Create basic script for booking a flight

Step3: Create standard checkpoint and parameterize the checkpoint with expected values

Navigation:

1. Select position in the script
2. Enter into recording mode
3. Goto insert menu
4. Select checkpoint- standard checkpoint
5. Move hand icon on to update order button and click on it
6. Click on ok after verification in object selection property window
7. Choose parameter option
8. Click on parameter options button
9. Select column name of database
10. Click on ok
11. Click on ok

Generated Script:

Window("Flight Reservation").Activate

Window("Flight Reservation").WinMenu("Menu").select "File New Order"

Window("Flight Reservation").ActivateX("MaskEdBox").Type "092618"

Window("Flight Reservation").WinComboBox("Fly From").Select "Frankfruit"

Window("Flight Reservation").WinComboBox("Fly From").Select "Los Angeles"

Window("Flight Reservation").WinButton("Flight").Click

Window("Flight Reservation").Dialog("Flights table").WinList("From").select "20332 FRA 10:12 AM LAX 05:23 PM AA $112.2.0"

Window("Flight Reservation").Dialog("Flights Table").WinButton("Ok").click

Window("Flight Reservation")..WinEdit("Name").Set "RGMCET"

Window("Flight Reservation").WinRadioButton("First").Set

Window("Flight Reservation").WinButton("Insert Order").Click

Window("Flight Reservation").WinButton("Update ORder")check.CheckForm("Update Order")

Test Results:

Test Result Summary

Iterations	Summary
1	Passed
2	Passed
3	Passed
4	Passed

Status	Times
Passed	4
Failed	0
Warnings	0

Details:

Update Order Results	
Property Name	**Property Value**
Enabled	True

Update Order Results	
Property Name	**Property Value**
Enabled	False

Update Order Results	
Property Name	**Property Value**
Enabled	True

Update Order Results	
Property Name	**Property Value**
Enabled	False

Text CheckPoint:

Text checkpoint is used to verify text field content without expected value

Create basic script to verify 5th order number record customer name as RGMCET in *Name* edit box?

Test Scenario:

Step1 : create basic script for opening 5th order number record

Step2: create text checkpoint to verify the customer name as RGMCET in *Name* edit box

Navigation:

1. Select position in the script
2. Enter into recording mode
3. Goto insert menu
4. Select checkpoint- text checkpoint
5. Move hand icon on to the corresponding element and click on it
6. Click on ok after confirmation in object selection properties window
7. Enter expected value in the constant option
8. Click on ok

Generated Script:

Window("Flight Reservation").Activate

Window("Flight Reservation").WinMenu("Menu").select "File; Open Order…"

Window("Flight Reservation").Dialog("Open Order").WinEdit("Edit").set "5"

Window("Flight Reservation").Dialog("Open Order").WinButton("ok").click

Window("Flight Reservation").WinEdit("Name: ")Check CheckPoint("Name: ")

Test Results:

Test Result Summary

Iterations	Summary
1	Failed

Status	Times
Passed	0
Failed	1
Warnings	0

Details :

Text checkpoint captured "RGM" expected "RGMCET"

TextArea Checkpoint:

Using this checkpoint we can verify selected portion of text in AUT with our expected value

Create basic script to verify the agent name as "RGM" in the graph window

Test Scenario:

Step1 : create basic script to open graph window

Step2: create textarea checkpoint to verify agent name as “RGM”

Navigation:

1. Select position in the script
2. Goto insert menu
3. Select checkpoint- Textarea checkpoint
4. Select portion of text by moving cross hair pointer on text and click
5. Click on ok after verification in object selection properties window
6. Enter expected value in the constant option
7. Click on ok

Generated Script:

Window(“Flight Reservation”).Activate

Window(“Flight Reservation”).WinMenu(“Menu”).select “Analysis graphs”

Window(“Flight Reservation”).Dialog(“Graph”).Resize 359.554

Window(“Flight Reservation”).Dialog(“Graph”).Move 621.39

Window(“Flight Reservation”).Dialog(“Graph”).Active(“pinnacle-BPS Graph”)

Window(“Flight Reservation”).WinObject(“GS_Drawing”).Check CheckPoint(“Gs_Drawing”)

Test Results:

Test Result Summary

Iterations	Summary
1	Failed

Status	Times
Passed	0
Failed	1
Warnings	0

Details :
Text checkpoint captured “RGMCET” expected “RGM” Match case : OFF Exact match : OFF Ignore spaces : ON

Bitmap Checkpoint:

Using this checkpoint we can compare image objects in different versions of applications.

Create basic script to compare image in MS-Paint window before and after modification using bimap checkpoint

Test Scenario:

Step1: Open MS-Paint window & draw an image (without recording)

Step2: Create bitmap checkpoint & provide image in paint window as expected image

Step3: Modify the image (without recording) & run the bitmap checkpoint

Navigation:

1. Enter into recording mode
2. Goto insert menu
3. Select checkpoint- bitmap checkpoint
4. Click on the image in MS-Paint window
5. Click on ok after confirmation in object selection property window
6. Click on ok after verifying image

Generated script:

Window(“Paint”).WinObject(“AFX:1000000:8”).Check CheckPoint(“AFX:1000000:8”)

Test Results:

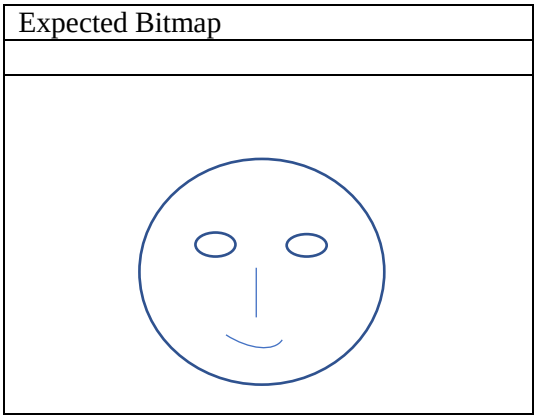
Test Result Summary

Iterations	Summary
1	Failed

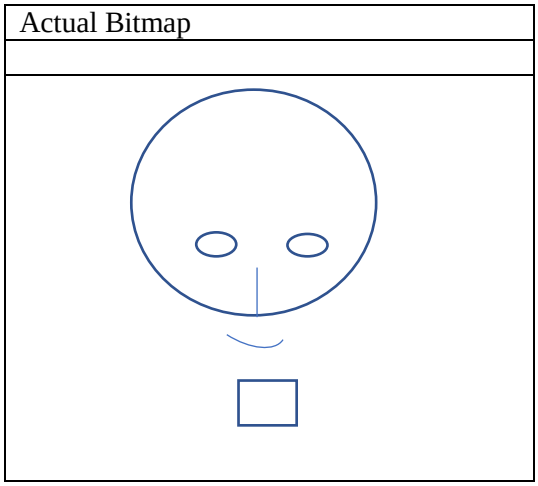
Status	Times
Passed	0
Failed	1
Warnings	0

Bitmap CheckPoint:

Expected Bitmap



Actual Bitmap



Database CheckPoint:

Using this checkpoint we can verify database content with respect to the operation performed on application frontend.

Create database checkpoint to verify 5th order number customer name as "RGMCET" in orders database table?

Test Scenario:

Step1: Create database checkpoint by providing 5th order customer name as "RGMCET" in orders database table.

Navigation:

1. Enter into recording mode
2. Goto insert menu
3. Select checkpoint- DataBase checkpoint
4. Select “specify SQL statement manually” option
5. Click on next
6. Select Machine DataSource tab
7. Select current project Database name (QT_Flight 32)
8. Click on ok
9. Write SQL statement manually
10. Click on finish
11. Enter expected value as “RGMCET” in constant option
12. Click on ok

Generated Script:

DbTable(“DbTable”).Check CheckPoint(“DbTable”)

Test Results:

Test Result Summary

Iterations	Summary
1	Failed

Status	Times
Passed	0
Failed	1
Warnings	0

Standard CheckPoint “DbTable” :Failed

Details

Verification type : String Content Settings:
Exact match- ON : Ignore space-ON : Match
Case-OFF. Results : checked 1 cells : successed : 0
Failed : 1

	1
1	XRGMCET

XML CheckPoint:

Using this checkpoint we can verify XML file content.

Create XML checkpoint to verify *student name* as “Ashish” and *student id* as “343” in college XML file.

Test Scenario:

Step1: create XML file for college XML

```
<?xml version = ”1.0”>
<college>
    <staff>
        <name> aaa </name>
        <id> 123 </id>
    </staff>
    <student>
        <name> bbb </name>
        <id> 555 </id>
    </student>
</college>
```

Step2: Create XML CheckPoint to verify name “Ashish” and id is “343”

Navigation:

1. Enter into recording mode
2. Goto insert menu
3. Select checkpoint- XML CheckPoint
4. Browse for xml file
5. Click on ok
6. Select values and enter expected values
7. Click on ok

Generated Script:

XMLfile(“clg.xml”).Check CehckPoint(“clg.xml”)

Test Results:

Test Result Summary		
XML CheckPoint		
XML validation : FAILED XML CheckPint type : File View XML CehckPoint Results…		
XML CheckPoint Result		
	Expected XML Tree	**Actual XML Tree**
	-college -staff -name -aaa -id -123 -student -name -bbb -id -555	-college -staff -name -aaa -id -123 -student -name -Ashish -id -343

Check Status Expected Actual
➢ This element was not checked

Accessibility CheckPoint:

Using this checkpoint we can verify **W3C** schools on the webapage.

Image CheckPoint:

Using this checkpoint we can verify image – object properties on a webpage.

Table CheckPoint:

Using this checkpoint we can verify webpage tables content.

Page CheckPoint:

Using this checkpoint we can verify webpage properties like lowtime, no.of images, no.of link etc.,

12. Output Values

Standard Output value

Output values:

Output values are used to read actual values from AUT during runtime. There are 5 types of output values:

1. Standard Output value
2. Text Output value
3. Text area Output value
4. Database Output value
5. XML Output value

Standard Output value:

Using this option we can read standard property values of an application object during runtime.

Create basic script to read items count property value from flyfrom listbox of flight Reservation application

Test Scenario:

Create standard output value to read items count property value from flyfrom listbox.

Navigation:

1. Enter into recording mode.
2. Goto insert menu
3. Select output value – Standard output value.
4. Move the handicon onto flyfrom listbox & click on it.
5. Click on OK after confirmation in object selection properties window.
6. Select items count property.
7. Click on modify button.
8. Enter column name.
9. Click on OK
10. Click on OK

Generated Script:

Window(“FlightReservation”).winComboBox(“FlyFrom”).OutputCheckPoint(“FlyFrom”)

Test Result Summary

Iterations	Summary
1	Done

Status	Times
Passed	1
Failed	0
Warnings	0

Runtime Datatable:

A1	10				
	Itemscount	B	C	D	E
1	10				
2					
3					
4					
5					
6					

Standard Output value

Create a basic script to verify delete ordered button enabled property WRT update order button enabled property after opening a record in Flight Reservation Application?

Test Scenario:

1. Create basic script for opening a record.
2. Create standard output value to read update order button enabled property value.
3. Create standard checkpoint to verify delete order button enabled property by parameterizing checkpoint.

Navigation:

1. Enter into recording mode.
2. Goto insert menu
3. Select output value-standard output value.
4. Move handicon on to update order button & click on it.
5. Click on OK after confirmation in object selection property window.
6. Select enabled property
7. Click on the modify button
8. Enter Column name
9. Click on OK.
10. Click on OK

Generated Script:

Window("FlightReservation").Activate

Window("FlightReservation").WinMenu("Menu").Select "File; Open Order…"

Window("FlightReservation").Dialog("OpenOrder").WinCheckBox("OrderNo").Set "ON"

Window("FlightReservation").Dialog("OpenOrder").WinEdit("Edit").Set "5".

Window("FlightReservation").Dialog("OpenOrder").WinButton("OK").Click

Window("FlightReservation").WinButton("UpdateOrder").OutputCheckPoint("UpdateOrder")

Window("FlightReservation").WinButton("DeleteOrder").CheckCheckPoint("DeleteOrder")

Test Results:

Test Result Summary

Iterations	Summary
1	Passed

Status	Times
Passed	1
Failed	0
Warnings	0

Runtime Datatable:

A1	"True"				
	Enabled	Denabled	C	D	E
1	True	True			
2					
3					
4					

Details:

Delete Order Details:

Property Name	**Property Value**
Enabled	True

Text Output Value

Text Output Value:

Using this option we can read text field content during runtime

Create basic script to verify customer name facts order window WRT existing name in the edit box in flight reservation window?

Test Scenario:

1. Create basic script to open 5th order no record
2. Create text output value to read customer name in the name edit box of fight reservation window.
3. Create basic script to open Facts order window.
4. Create Text checkpoint to verify the name field content in facts order window by parameterizing the checkpoint.

Navigation:

1. Enter into recoding name
2. Go to insert menu
3. Select output value test output value
4. Move the hand icon on to name edit box and click on it.
5. Click on the offer confirmation in object selection properties window
6. Click on modify
7. Enter column name
8. Click on OK
9. Click on OK

Generated Script:

Window("FlightReservation").Activate

Window("FlightReservation").WinMenu("Menu").Select "File; Open Order…"

Window("FlightReservation").Dialog("OpenOrder").WinCheckBox("OrderNo").Set "ON"

Window("FlightReservation").Dialog("OpenOrder").WinEdit("Edit").Set "5"

Window("FlightReservation").Dialog("OpenOrder").WinButton("OK").Click

Window("FlightReservation").WinEdit("Name:").OutputCheckPoint("Name")

Window("FlightReservation").Dialog("FaxOrder No 5.").Click 59-94

Window("FlightReservation").Dialog("FaxOrder No 5.").WinEdit("Name").Check Checkpoint("Name: z")

Output:

Test Result Summary

Iterations	Summary
1	Passed

Status	Times
Passed	1
Failed	0
Warnings	0

Runtime Datatable:

A1	"True"					
	Name	B	C	D	E	F
1	Kimsmith					
2						
3						
4						

Details:

Text Check Point Captured "Kimsmith"
Match case: OFF
Exact Match: OFF
Ignore Spaces: ON

Text area Output Value

Text area Output Value:

Using this option we can read selected portion of text from AUT during runtime.

Create a basic script to read total textfield content without $symbol after opening a record?

Test Scenario:

1. Create a basic to open 5th order no record.
2. Create textarea output value to react total textfield without $symbol.

Navigation:

1. Enter into the recording mode.
2. Go to insert menu
3. Select output value- Text area output value.
4. Move crosshair pointer on to total text field & select value without $
5. Click on modify button in output values windows.
6. Enter column name
7. Click on OK
8. Click on OK

Generated Script:

Window("FlightReservation").Activate

Window("FlightReservation").WinMenu("Menu").Select "File; Open Order…"

Window("FlightReservation").Dialog("OpenOrder").WinCheckBox("OrderNo:").Set "ON"

Window("FlightReservation").Dialog("OpenOrder").WinEdit("Edit").Set "5"

Window("FlightReservation").Dialog("OpenOrder").WinButton("Ok").Click

Window("FlightReservation").WinEdit("Total").OutputCheckPoint("Total")

Test Results:

Test Result Summary

Iterations	Summary
1	Done

Status	Times
Passed	1
Failed	0
Warnings	0

Runtime Data table:

A1	Kimsmith
A1	3176.46
	Total B C D E F
1	3176.46
2	
3	
4	

Database Output Value

Text area Output Value:

Using this option we can read database content during runtime.

Create a basic script to read 5th order no customer name from the orders database table?

Test Scenario:

1. Create database output value to read 5th order no record customer name from orders database table.

Navigation:

2. Enter into the recording mode.
3. Go to insert menu
4. Select output value- Database output value.
5. Select "specify SQL statement manually" option in database query.
6. Click on next
7. Click on create button
8. Click on Machine datasource tab.
9. Select current project datasource name(QT_FLIGHT_32).
10. Click on OK
11. Write SQL statement manually
12. Click on Finish
13. Select customer name from order and click on add to output value button
14. Click on modify
15. Click on column name
16. Click on OK
17. Click on OK

Generated Script:

```
Window("Dbtable").Output CheckPoint("Dbtable")
```

Test Results:

Test Result Summary

Iterations	Summary
1	Done

Status	Times
Passed	1
Failed	0
Warnings	0

A1	Kimsmith
	Name B C D E
1	Kimsmith
2	
3	
4	

Details:

Results: Checked 1 cells : Succeeded 1: Failed 0

XML Output Value

XML Output Value:

Using this option we can read database content during runtime.

Create a basic script to read student name& id from college XML file?

Test Scenario:

College XML have to be created for college information

<?xml version=”1.0”>

<college>

<staff><name>aaa</name>

<id>530</id></staff>

<student><name>bbb</name>

<id>555</id></student></college>

Create XML output value to read student name & if from college XML file.

Navigation:

1. Enter into recording mode
2. Go to insert menu
3. Select output value – XML output value
4. Browse for XML file
5. Expand for XML file
6. Student name value & student id values are selected
7. Give column names
8. Click on OK
9. Click on OK

Generated Script:

Window(“Dbtable”).Output CheckPoint(“Dbtable”)

Test Results:

Test Result Summary

Iterations	Summary
1	Done

Status	Times
Passed	1
Failed	0
Warnings	0

A1	119					
	Name	id	C	D	E	F
1	Swapni	119				
2						
3						
4						
5						
6						
7						

Write a script to verify the no.of items flyfrom listbox with expected output as 8?

Test Scenario:

Add the objects to flight reservation application into object repository

Write the script manually to verify the no.of items from flyfrom listbox with expected output as 8

Navigation:

- Goto resource menu
- Select object repository
- Goto object menu in object repository window
- Select add objects to local option
- Click on flight reservation
- Click on ok after confirmation in object selection property window
- Select option “All object types”
- Click on ok
- Close object repository

Script:

```
n=Window(“FlightReservation”).WinComboBox(“FlyFrom”).GetItemsCount

If n==9 then

        Reporter.ReportEvent micPass, ”Items Count Validation”, “Expected count”

Else
```

Reporter.ReportEvent micFail, ”Items Count Validation”, “Actual count”

Is : &space(4)&n

EndIf

Test Result Summary

Iterations	Summary
1	Failed

Status	Times
Passed	0
Failed	0
Warnings	0

StepName: ItemsCountValidation

Step Failed

Object ItemsCount Validation	Actual Count is 10	Failed	10/04/2018 15:30:24

Script to get List Box items

Write the script manually to print all fly from list box items with index

Test Sceario:

Add all test objects from flight reservation into object repository

Write the script manually to print all fly from list box items with index

Navigation:

- Goto resource menu
- Select object repository
- Goto object menu in object repository window
- Select add objects to all local option
- Click on flight reservation
- Click on ok after confirmation in object selection property window
- Select option “All object types”
- Click on ok
- Close object repository

Script:

```
n=Window(“FlightReservation”).WinComboBox(“FlyFrom”).GetItemsCount

For i=0 to n-1

        x=window(“FlightReservation”).WinComboBox(“FlyFrom”).GetItem(i)

        print i&space(4) “Index item is: “&space(4)& x

Next
```

Test Result Summary

Iterations	Summary
2	Failed

Status	Times
Passed	0
Failed	0
Warnings	0

Quick Test Print log

File

```
0 Index item is: Denver
1 Index item is: Paris
2 Index item is: Switzerland
3 Index item is: Berlin
4 Index item is: Tokyo
5 Index item is: San Francisco
6 Index item is: Seattle
7 Index item is: Sydney
8 Index item is: Nairobi
```

13. Object Repository:

Object Repository is a place where we can maintain required test objects with respect to script with those objects object repository acts as an interface between script to application objects at runtime

Operations in Object Repository:

- ***Adding test objects:*** To identify runtime objects in AUT we need to add required test objects in object repository
- ***Delete test objects:*** Sometime we can delete test objects from the object repository

Output Function:

1. **MessageBox():** This function displays user message in popup window and execution will pass at that statement until we close popup window

   Syntax : msgBox(user msg)

2. **Print():** It displays user message in a quick test print log window and execution will continue

   Syntax : print(user msg)

3. **ReportEvent():** Using these method we can pass the user message into QTP test result window along with step status

   Syntax : Reporter.ReportEvent stepstatus, stepname, step description

Methods:

getItemsCount(): using this method we can find number of times available in list box

getItem(): using this method we can read listbox items based on index

14. Writing Script Manually

Test Script Template is a reusable formatted document that contains pre-selected information important for creating a usable test script. This document determines how detailed your tests are and what information should be included in each test case.

- Test Scripts means a line-by-line description containing the information about the system transactions that should be performed to validate the application or system under test.
- Test case is a step by step procedure that is used to test an application whereas the test script is a set of instructions to test an application automatically.
- Three ways to create test script are
- 1) Record/playback 2) Keyword/data-driven scripting, 3) Writing Code Using the Programming Language.
- Your test script should be clear and you should create a test script that should contain just one specific action for testers to take.
- Using a test script is the most reliable approach to verify that nothing is skipped and that the results are true as the desire testing plan.
- Test Script Template is a reusable formatted document that contains pre-selected information important for creating a usable test script.
- Test Scripts are a line-by-line description of all the actions that are necessary to perform and test on specific user journeys. It lists out each step that should be taken with the expected results. Then testers can easily as systematically test each step on a wide range of devices. Testing according to a prescribed test steps is known as scenario Testing.
- The test script approach has specific advantages. It leaves a lot less room for error during the testing process. Sometimes, when testers are left up to freely browse through the product, they can miss certain features or assume that a function has the expected result when it, in fact, does not. Using test scripts is the most reliable approach to verify that nothing is skipped over and that the results are truly as the publisher wants them.
- The test script approach is particularly useful if the sequence of the user performance is important and specific. For example, if the user is not supposed to have very much freedom and are supposed to take a very specific user journey, such as creating an account or ordering a specific product.

- If the test script is not clear, testers will have to constantly ask the project manager the details about the instructions, which wastes time and interrupt productivity. Verifying that each step in the test script is clear, concise, and coherent will keep the testing process rolling at its proper pace.

  Simple

- Each step in the test script should contain just one specific action for testers to take. This ensures that each function is tested correctly and that testers don't accidentally pass by anything on the user path.

  Well-thought-out

- To write the test script, script writers should put themselves in the place of the user to decide which user paths to test. They should be creative to be able to predict all of the different paths that users would take.

- Properly creating test scripts saves time during the testing process and ensures a higher quality product. Script writing and testing requires experience to be able to know where bugs hide out, to be familiar with user paths, and to list out each step with clarity.

15. Study of Rational Functional Tester (RFT)

What is IBM Rational Functional Tester?
IBM Rational Functional Tester is an automated functional testing and regression testing tool. This software provides automated testing capabilities for functional, regression, GUI and data-driven testing. It supports a range of applications, such as web-based, .Net, Java, Siebel, SAP, terminal emulator-based applications, PowerBuilder, Ajax, Adobe Flex, Dojo Toolkit, GEF, Adobe PDF documents, zSeries, iSeries and pSeries.

Benefits
Storyboard testing

Simplifies test visualization and editing using natural language and rendered screenshots.
Automated testing
Enables testers to automate tests resilient to frequent application user interface changes with ScriptAssure technology.

Data-driven testing

Let's you perform the same series of test actions with a varying set of test data.
Test scripting
Combines a recorder of user actions with multiple customization options and intelligent script maintenance capabilities.

Integrates with other software

Integrates with IBM Rational Team Concert and IBM Rational Quality Manager to provide access to work items and logical or compound SCM test asset support.

Supports team collaboration
Supports sharing of functional tests across team members and running on hybrid environments with integrations with Rational Test Automation Server.

Key features

- Visual editing through screenshots
- Advanced Script Assure technology
- Earlier data detection
- Test scripting
- Integration with other software
- Mobile native testing
- Guided healing for test cases
- Supports all browsers

Features

Visual editing through screenshots

IBM Rational Functional Tester provides a visual storyboard format for representing test actions. Storyboard testing combines natural language test narrative with visual editing through application screenshots. That means novice and professional testers can communicate and understand test flow and edit test actions without reading or writing test script code.

Advanced Script Assure technology

With the advanced Script Assure technology, this software allows you to accommodate frequent user interface changes and avoid increases in maintenance overhead. Script Assure uses fuzzy matching algorithms to locate objects during test execution, even if the objects have changed since test creation. You can develop automation scripts that are associated with keywords. Keyword testing promotes script reuse and enables manual testers to use automation within manual test cycles.

Earlier data detection

Use IBM Rational Functional Tester to automatically detect data entered during test recording and prepare the test for data-driven testing. Using a spreadsheet-like data editor, you can create customized data sets to be used by the test during playback. This software manages validation of dynamic data with multiple verification points and support for regular expression pattern matching.

Test scripting

Choose between either Java or Visual Basic .NET; testers using Java can work in the Eclipse Java editor, and those using Visual Basic .NET can work in Visual Studio .NET. This software offers scripting to create, edit and execute tests on the Linux platform, including everything except a test recorder. On the Windows platform, the software provides all recording, editing and execution capabilities. It supports automated version control to maintain multiple test sets and to enable parallel development.

Integration with other software

IBM Rational Functional Tester provides IBM® Jazz® integration to support collaborative application lifecycle management. IBM Jazz Eclipse Client integration provides IBM Rational Functional Tester access to work items within IBM Rational Team Concert and IBM Rational Quality Manager. Enhanced SCM integration with IBM Rational Team Concert supports the management and sharing of test assets.

16. Study of Selenium

Brief Introduction Selenium IDE

Selenium Integrated Development Environment (IDE) is the **simplest framework** in the Selenium suite and is **the easiest one to learn**. It is a **Firefox plugin** that you can install as easily as you can with other plugins. However, because of its simplicity, Selenium IDE should only be used as a **prototyping tool**. If you want to create more advanced test cases, you will need to use either Selenium RC or WebDriver.

Selenium Grid

Selenium Grid is a tool **used together with Selenium RC to run parallel tests** across different machines and different browsers all at the same time. Parallel execution means running multiple tests at once.

Features:

- Enables **simultaneous running of tests** in **multiple browsers and environments.**
- **Saves time** enormously.
- Utilizes the **hub-and-nodes** concept. The hub acts as a central source of Selenium commands to each node connected to it.

Brief Introduction Selenium Remote Control (Selenium RC)

Selenium RC was the **flagship testing framework** of the whole Selenium project for a long time. This is the first automated web testing tool that **allowed users to use a programming language they prefer**. As of version 2.25.0, RC can support the following programming languages:

- Java
- C#
- PHP
- Python
- Perl
- Ruby

Write the working process of selenium?

Selenium is a type of plugin used in Firefox browser. It is used to record user actions on web application

Selenium working process:

1. Open Selenium IDE from tools menu in firefox browser by default the record button is ON
2. Open web application on which we need to record and perform any user actions
3. Record while performing use actions
4. Click on record button to stop recording
5. To run the test script we have 2 steps
    - Play current test case
    - Play entire test suit
6. Selenium IDE displays the recorded script in two formats
    - Table format
    - Time source code
7. In Selenium IDE the script will run only in HTML format

Evaluation Procedure for Internal Laboratory Examinations:

1. Of the 25 marks for internal, 10 marks will be awarded for day-to-day work and 10 marks to be awarded for the Record work and 5 marks to be awarded by conducting an internal laboratory test.
2. Concerned Teachers have to do necessary corrections with explanations.
3. Concerned Lab teachers should enter marks in index page.
4. Internal exam will be conducted by two Staff members.

Dr.K. Subba Reddy

Professor & Head Dept. of CSE.

Evaluation Procedure for External Laboratory Examinations:

1. For Practical subjects there is a continuous evaluation during the semester for 25 Sessional marks and 50 end examination marks.
2. The end examination shall be conducted by the teacher concerned (Internal Examiner) and another External Examiner, recommended by Head of the Department with the approval of principal.

Evaluation procedure for external lab examination:

1. Procedure for the program	---------- 20M
2. Execution of the program	---------- 15M
3. Viva voce	---------- 15M

Total	50M

Dr.K. Subba Reddy
Professor & Head Dept. of CSE.