Week-1

1). Aim: Program in 'c' language to demonstrate the working of the following constructs

i) do-while ii) while-do iii) if else iv) switch v) for

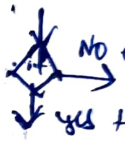
Path testing

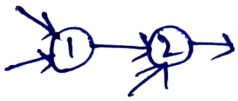
Theory: 2 kinds of basic program statements.

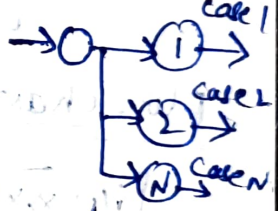
control flow graphs: A simplified abstract and graphical representation of a program's control structure using process blocks, decissions and junctions.

symbols:

Process block → 

→ Decissions 

junctions 

→ Case statement 

Control Flow graph Elements:

process Block: A sequence of program statements uninterrupted by decissions or junctions with a single entry and exit.

Junction: A point in the program where control flow can merge. Ex: target of goto, jump, continue.

Decissions: A program point at which control flow diverge. Eg: if statement.

Case statements: A multiway branch or decission.

Ex: In assembly language jump addreses table, multiple gotos, case/switch.

For test design case stmt and decissions are similar.

Control flow graph vs Flow charts

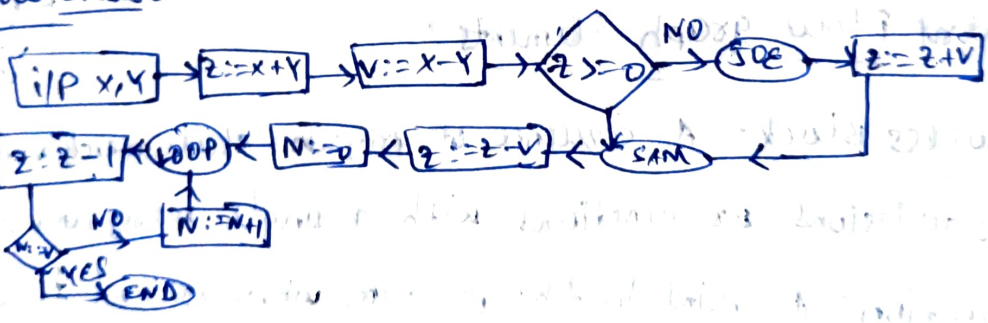
- → compact representation of program
- → focus on i/p & o/p & the control flow into and out of block.
- → inside details of a process block are not shown

- → Usually a multi page description.
- → focus on process steps inside.
- → Every part of the process block are drawn.

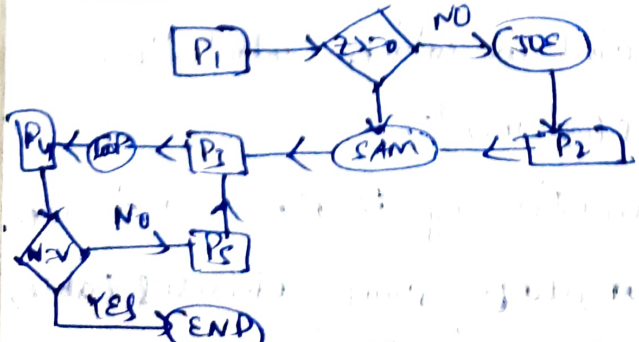
Eg:

```
INPUT X,Y
Z := X + Y
V := X - Y
IF Z >= 0 GOTO SAM
JOE : Z := Z + V
SAM : Z := Z - V
      FOR N = 0 to V
      Z : Z - 1
      NEXT N
      END
```

Flow chart



Flow graph



Simplified flow graph

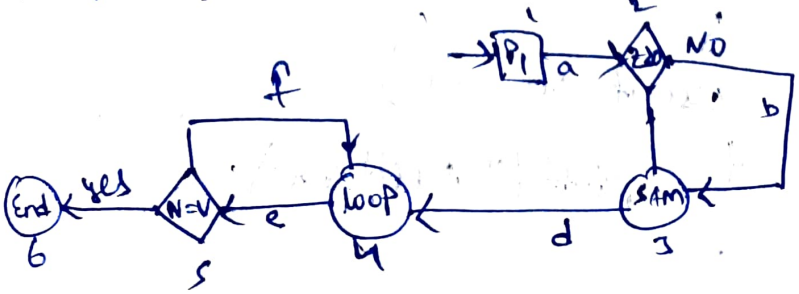
path testing criteria

1. Path Testing (P_n) : Execute all possible control flows. paths in the program, but typically restricted to entry-exit paths implies 100% path coverage. impossible to achieve.

2. statement testing (P_1) : execute all statements in program atleast once under some test. 100% statement coverage ⇒ 100% node coverage.

3. Branch testing (P_2) : Execute enough tests to assure that every branch alternative has been exercised atleast once under some test.
denoted by C_2.
objective : 100% branch coverage and 100% link coverage
for well structured s/w branch & coverage testing include stmt coverage.

code :
procedure to perform path testing
step-1 : write a code which includes if, while, do while
step-2 : convert code to flow graph.
step-3 : give appropriate numbering to flow graph
step-4 : perform stmt coverage, branch coverage, path coverage.
step-5 : conclude with output.



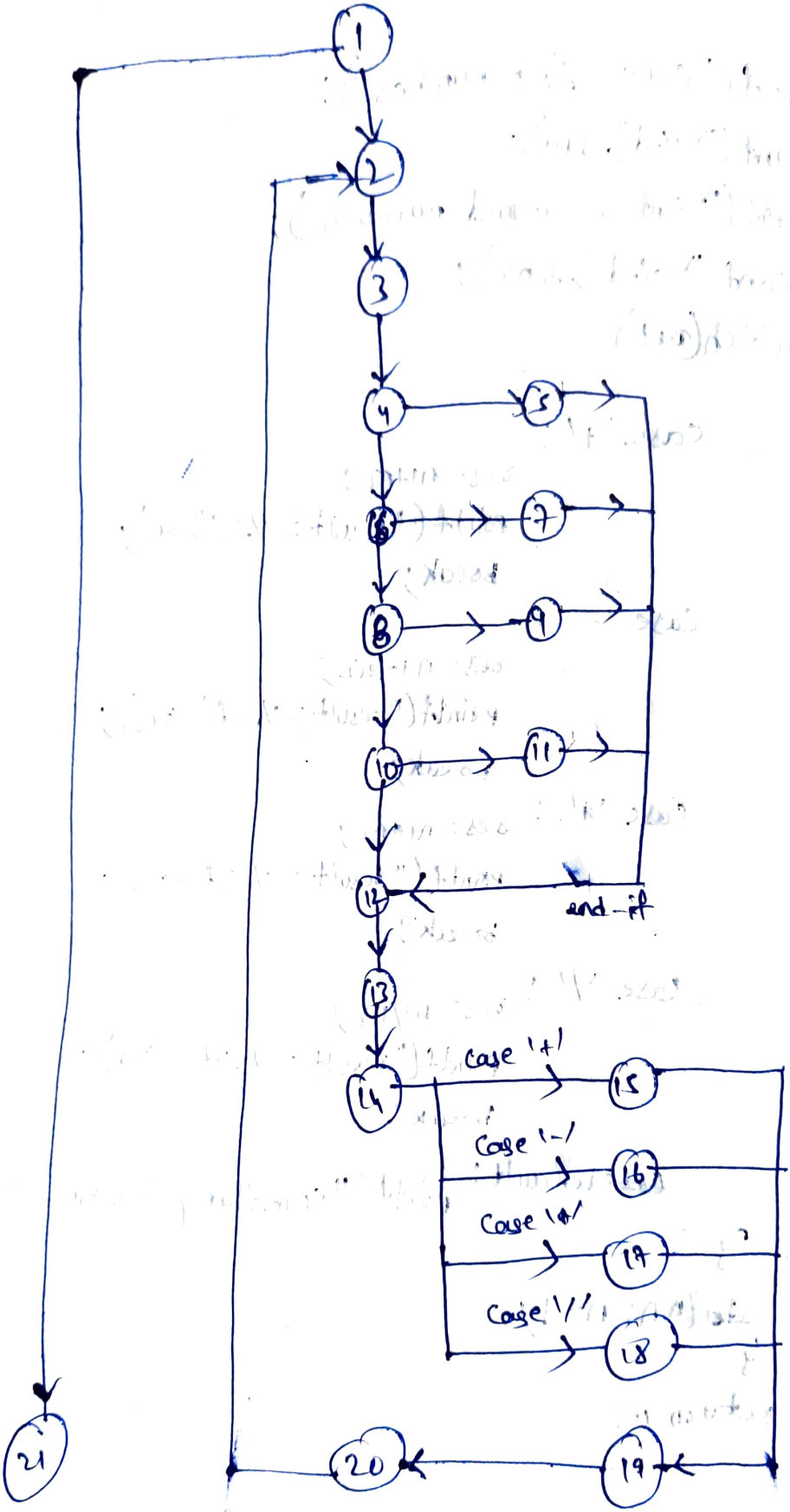
Paths	Decisions		Process link						
	2	5	a	b	c	d	e	f	g
abdeg	Y	Y	Y	Y	Y	Y	Y	Y	Y
acdeg	N	Y	Y	Y	Y	Y	Y	Y	Y
abdefeg	Y	N	Y	Y	Y	Y	Y	Y	Y
acdefeg	N	Y	Y	Y	Y	Y	Y	Y	Y

Program:

```
#include<stdio.h>
int main()
{
    int n,n1,n2;
    char opt;
    float res;
    printf("enter no of operations");
    scanf("%d",&n);
    while(n)
    {
        printf("choose an operator(+,-,*,/)");
        scanf("%c",&opt);
        if(opt=='/')
        {
            printf("you selected division");
        }
        else if(opt=="*")
        {
            printf("multiplication");
        }
        else if(opt=='+')
        {
            printf("Addition");
        }
```

```
else if(opt=='-')
{
    printf("subtraction");
}
printf("Enter first number");
scanf("%d",&n1);
printf("Enter second number");
scanf("%d",&n2);
switch(opt)
{
    case '+':
        res=n1+n2;
        printf("result = %.2f", res);
        break;
    case '-':
        res=n1-n2;
        printf("result = %.2f", res);
        break;
    case '*': res=n1*n2;
        printf("result = %.2f", res);
        break;
    case '/': res=n1/n2;
        printf("result = %.2f", res);
        break;
    default: printf("something is wrong");
}
n=n-1;
}
return 0;
}
```

Control flow graph:



Output:

statement & branch coverage:

Test case 1:

4, /

sc path 1: 1-2(T)-3-4(T)-5-12-13-14-18-19-20-2(F) 21

Test case 2:

1, a

sc path 2: 1-2(T)-3-4(F)-6(T)-7-12-13-14-17-19-20 2(F) 21

Test case 3:

1, +

sc path 3: 1-2(T)-3-4(F)-6(F)-8(T)-9-12-13-14-15-19-20-2(F)-21

Test case 4:

1, -

sc path 4: 1-2(T)-3-4(F)-6(F)-8(F)-10(T)-11-12-13-14-16-19-20-2(F)-21

Path coverage:

Test case 1:

0 or a or b

Path 1: 1-2(F)-21

Test case 2:

0 1, +

path 2: 1-2(T)-3-4(T)-5-12-13-14-18-19-20-2(F)-21

Test case 3:

1, *

Path 3: 1-2(T)-3-4(F)-6(T)-7-12-13-14-17-19-20-2(F)-21

Test case-4:

1, +

Path 4: 1-2(T)-3-4(F)-6(F)-8(T)-9-12-13-14-15-19-20--2(F)-21

Test case 5:

1, -

Path 5: 1-2(T)-3-4(F)-6(F)-8(F)-10(T)-11-12-13-14-16-19-20-2(F)-21

2)

Aim: 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.

Theory:

Validation Testing:

- It succeeds when the software functions in a manner that can be reasonably expected by the customer.
- Reasonable expectations are defined in the SRS, the specification contains a section called validation criteria, information contained in that section forms the basis for a validation testing.

1) Validation Test criteria:

After each validation test case has been conducted, one or two

possible conditions exists

1. The function or performance characteristic confirms to specification and is accepted.
2. A deviation from specification is uncovered and a deficiency list is created.
3. Deviation or error discovered at this stage in a project can rarely be correct prior to scheduled delivery.

2) Configuration Review:

The intent of review is to ensure that all elements of the s/w configuration have been properly developed, are cataloged. sometimes called audit.

3) Alpha testing:

It is conducted at the developer's site by end-users.

- The s/w is used in a natural setting with the developes "looking over the shoulder" of typical users and recording errors and usage problems.
- Alpha tests, conducted in a controlled environment.

a) Beta testing:

It is conducted at the end-users site, unlike alpha testing, developes is generally not present.

- Beta test is a "live" application of the software in an environment that can't be controlled by the developes.
- End users records the problems and reports to the developes at regular intervals.
- As a result of problems reported during beta tests, s/w engineers make modifications and then prepare for release of the s/w product.

System Testing:

Its primary purpose is to test the complete software.

1) Recovery Testing:

many computer-based systems must recover from faults and resume processing within a prespecified time.

- Recovery testing is a system test that forces the software to fail in a variety of ways and verifies that recovery is properly performed.
- If recovery automatic (performed by the system itself), reinitialization, checkpointing mechanisms, data recovery and restart are evaluated for correctness.
- If recovery is human intervention, the mean-time-to-repair (MTTR) is evaluated to determine whether

it is within acceptable limits.

2). Security Testing:

Security testing verifies that protection mechanism built into a system infact protect it from improper penetration.

3) Stress testing:

Stress testing refers to a type of testing that is so harsh, it is expected to push the program to failure. For example, we might flood a web application with data, connections, and so on until it finally crashes. Buffer overflows are classic examples of stress test results.

4) Performance testing:

Performance testing is designed to test the run-time performance of software within the context of an integrated system.

- It occurs throughout all steps in the testing process, even at unit level also.
- Performance tests are often coupled with stress testing.

Program:

```
#include<stdio.h>
#define MAX 100
void main()
{
    int r1, c1, r2, c2, A[MAX][MAX], B[MAX][MAX],
        res[MAX][MAX], i, j, k;
    printf("Enter no of rows and no
        of columns of matrix A : ");
    scanf("%d%d", &r1, &c1);
    printf("Enter no of rows and no
        of columns of matrix B : ");
    scanf("%d%d", &r2, &c2);
    printf("Enter A array elements : \n");
```

```
for (i=0; i<r1; i++)
{
    for(j=0; j<c1; j++)
    {
        scanf("%d",&A[i][j]);
    }
}
printf("Enter B array elements : \n");
for(i=0; i<r2; i++)
{
    for (j=0; j<c2; j++)
    {
        scanf("%d",&B[i][j]);
    }
}
printf("AxB resultant matrix : \n");
for (i=0; i<r1; i++)
{
    for(j=0; j<c2; j++)
    {
        res[i][j]=0;
        for(k=0; k<c1; k++)
        {
            res[i][j] = res[i][j] + A[i][k]
                        * B[k][j];
        }
    }
}
```

```
    for(i=0; i<r1; i++)
    {
        for(j=0; j<c2; j++)
        {
            printf("%d", res[i][j]);
        }
        printf("\n");
    }
    getch();
}
```

Output:

causes of failures:

1) k undeclared
2) incompatible implicit declaration of built in function 'printf', 'scanf'

Week 1

week-2

1)

Aim: Take any system (eg: ATM system) and study its system specifications and report the various bugs.

Theory:

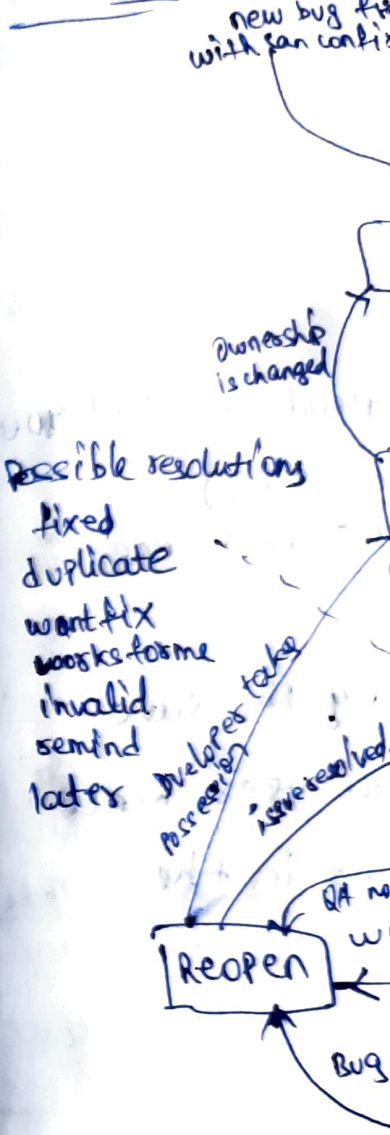
Bug definition: Deviates from expected behaviour is a bug.

Consequences:

- mild : misspelled output or mal-aligned printout
- moderate: o/ps are misleading or redundant impacting performance.
- Annoying:
- Disturbing: Legitimate transactions refused.
- serious: Losing track of transactions
- very serious: system does another transaction instead of requested.
- Extreme: frequent & arbitary
- Intolerable: long term unrecoverable corruption of db.

Catastrophic - system fails or shuts down.
Infectious: Corrupts other systems.

Bug life cycle:



Bug format:

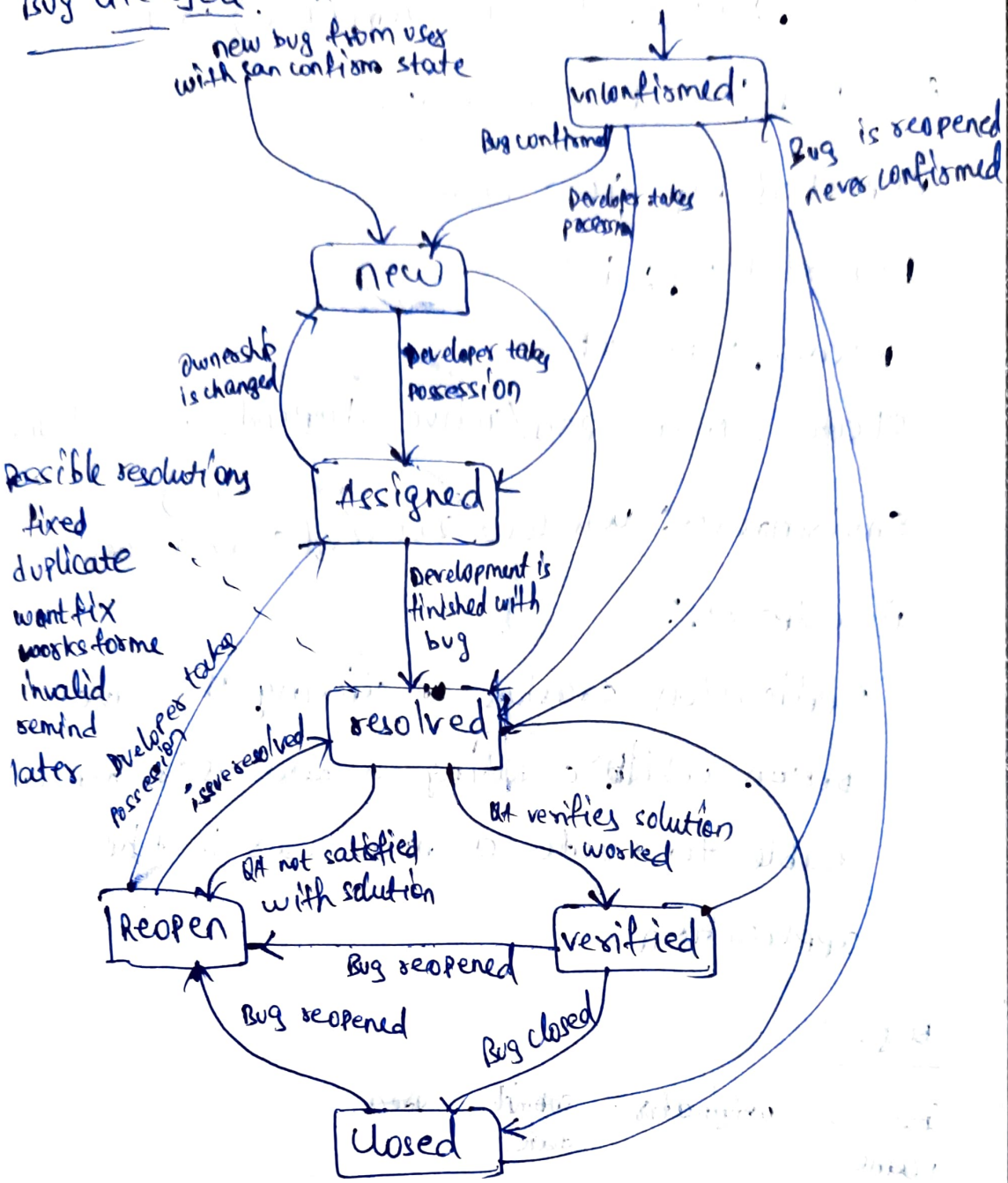
Bug Name:

Bug id:

Area pat

Build num

Bug life cycle:



Bug format:

Bug Name: Application crash on clicking save button while creating new users.

Bug id: automatically created by bug tracking tool.

Area path: users menu → New users

Build number: version number 5.0.1

Severity: High (High/medium/low) or ,

Assigned to: developer x

Reported by: Your Name

Reported on: date

Reason: Defect

Status: New/Open/active (depends on tool you are using)

Environment: window 2003/sql server 2005

Description:

Application crashes on clicking the save button while creating a new user, hence unable to create a new user in the application.

Bugs:

~~Issue~~ ~~Originator~~ ~~Submit~~ ~~Assd~~
~~Assd~~ date

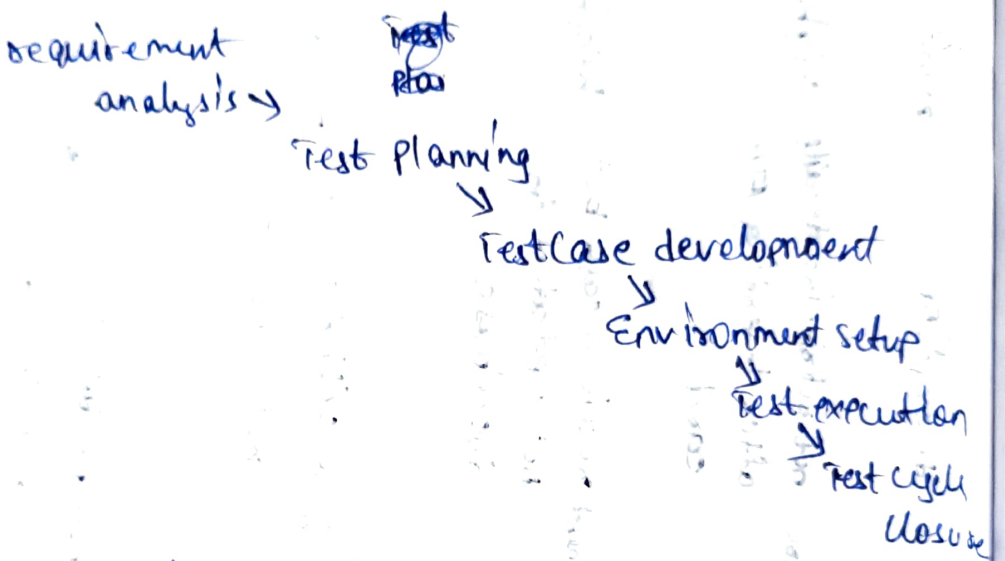
Issue name	originator	submit date	area	description	priority	state	Business impact	expected solution date	Assigned to
not able to read card	abc	1/11/21	hardware	not retrieving details after inserting ATM card	High	In progress	ATM services need to be paused	8/11/21	xyz
amount not debited from account	abc	1/11/21	development	received cash but not updated in database	high	In progress	database connected functionalities are to be paused	8/11/21	xyz
~~incor~~ wrong transaction history	abc	1/11/21	accounting	logs/history is not correct	medium	open		8/11/21	xyz
invalid amount	abc	1/11/21	development	not able to withdraw required cash	Low	open		8/11/21	xyz

4)

Aim: Write test cases for any known Application.
(Ex: Banking system, ATM)

Theory:

software testing life cycle:



Manual Testing: Testing when carried out without the help of any tool is called as manual testing.

Automation testing: Using automation tools to write and execute test cases is known as automation testing.

Benefits of automation testing :

- Reduction of repetitive work
- Repeatability
- Greater consistency
- Ease of access of information about tests or testing.
- Fast, repeatable, reusable, reliable, comprehensive, programmable.

Automation tools :

HP UFT : HP Unified Functional Testing is the market leader in functional testing tools. The tool supports plethora of environments including SAP, Java, Delphi amongst others. HP UFT can be used in conjunction with quality center.

Rational Robot : It is an IBM tool used to automate regression, functional and configuration tests for client severs, e-commerce as well as ERP applications. It can be used with Rational test manager.

Selenium: It is an open source web automation tool. It supports all types of web browsers.

Automation testing process

1. Learning
2. Recording
3. Edit script
4. Run
5. Analyze results

Manual Testing	vs	Automation testing
→ Time consuming & tedious		→ Fast
→ Huge investment in human resources		→ Less investment in human resources
→ Less reliable		→ more reliable
→ Non-programmable		→ Programmable

Testcase definition: A test case is step by step instructions to test a specific requirement. Step by step driving direction to check specific functionality.

→ One test case can contain one or more steps based on the complexity of requirement.

→ simple and clear steps or driving ~~directions~~ directions to test s/w functionality.

Test scenario: A document specifying a sequence of actions for execution of a test.

Test case: One or more input values, execution preconditions, steps for execution, expected results and execution post-conditions, developed for a particular objective or test condition.

Test data: Data that exists before a test is executed, and that affects or is affected by the component or system under test.

IEEE format for test case :

1. Test case id : Unique Name or ID

2. Test case name : Name of test condition

3. Features to be tested : Corresponding module/ Function / service.

4. Test suite Id : In which batch, is this case is a member of

5. Priority : Importance of test case in terms of functionality.

P0 : Basic functionality

P1 : General functionality (Input domain, Error handling, Compatibility)

P2 : Cosmetic testing (user Interface)

6. Test Environment : Required h/w & s/w to execute this test case.

7. Test Effort (Person/hour) : time required to execute.

8. Test Duration: Date & time (scheduling)

9. Test setup: Necessary tasks to do before starting this test case execution

10. Test procedure: A step by step process to execute these test cases from base state to end state.

11. Test case Pass/Fail criteria: when this test case is passed / failed.

Results:

Test scenario	Test case	Pre-conditions	Test Data	Expected result	Actual result	Pass/ Fail
check card details	check the card number validity after inserting card.					

Test Scenario	Test case	Pre-condition	Test Steps	Test data	Expected results	Actual results	Pass/Fail
Check card details validity	Check whether card number is valid.		Launch/On ATM insert card	Card/Act no: 1234567890	valid	valid	Pass
Pin validation	Check whether entered Pin is correct or not		Launch application insert card get account details	Pin number: 4567	valid	Invalid	Fail
Validate amount to be withdrawn	Check whether withdraw amt is less than or equal to available balance		Launch application insert card get account details get balance	Pin number: 4567 withdraw amt: 20,000	valid	valid	Pass