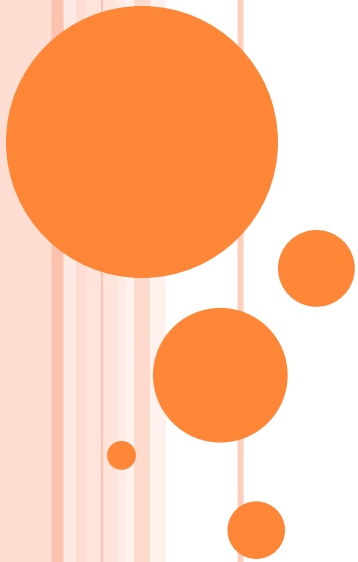# INTRODUCTION TO SOFTWARE TESTING

ABOUT THE COURSE

- Prescribed Textbook :
  - Aditya P Mathur *Foundations of Software Testing,* , Second Edition, Pearson Education, 2008.
  - Paul C. Jorgensen *Software Testing A Craftsman's Approach*, 3rd Edition, 2013.

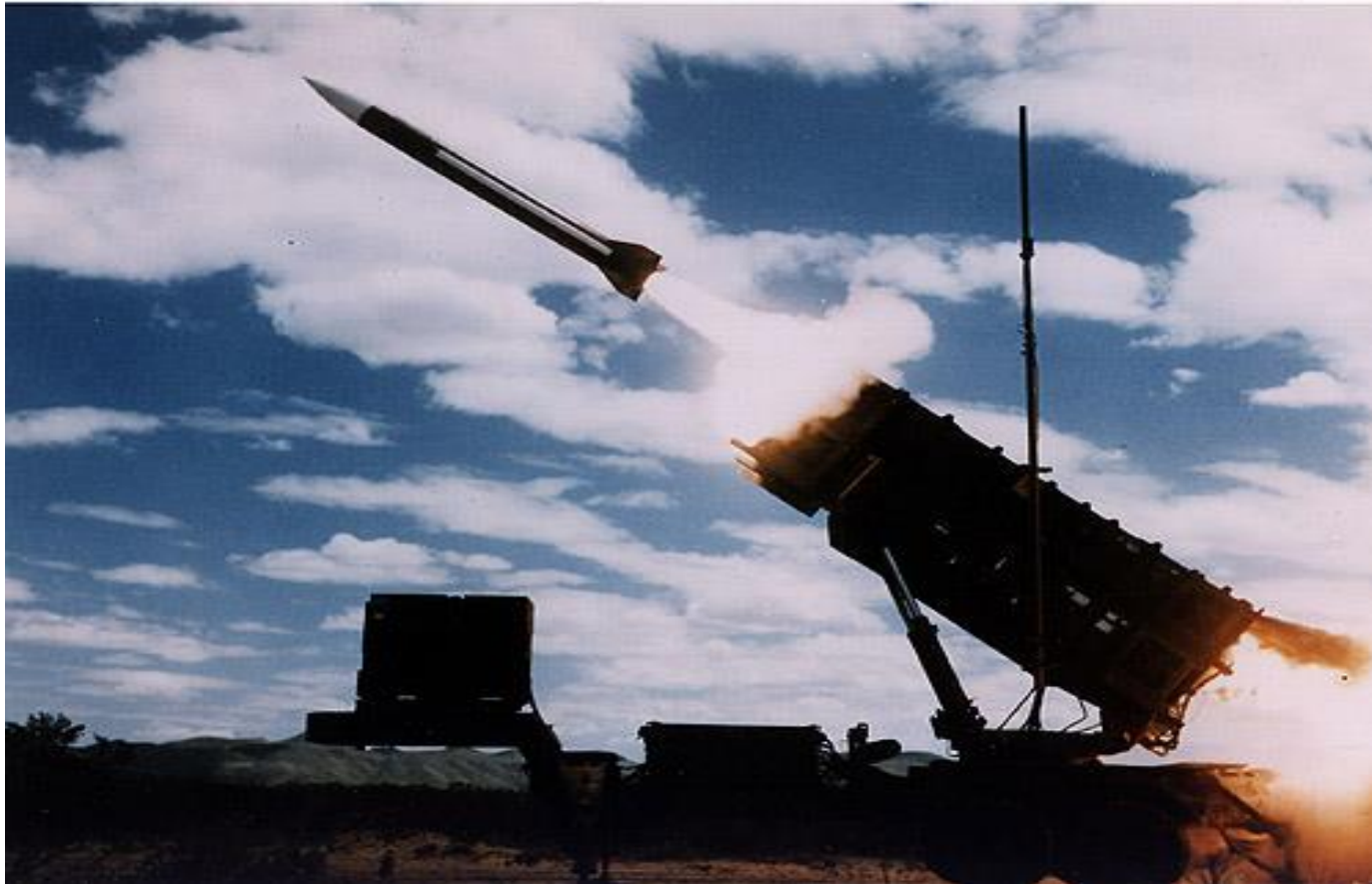- Maintain Separate class note.

INTRODUCTION

- What is software Testing?

- Why do we need to test software?

- Can we live without testing?

- How do we handle software bugs reported by the customer?

- SOFTWARE is not RELIABLE.

SOME SOFTWARE FAILURES

The Y2K problem

4
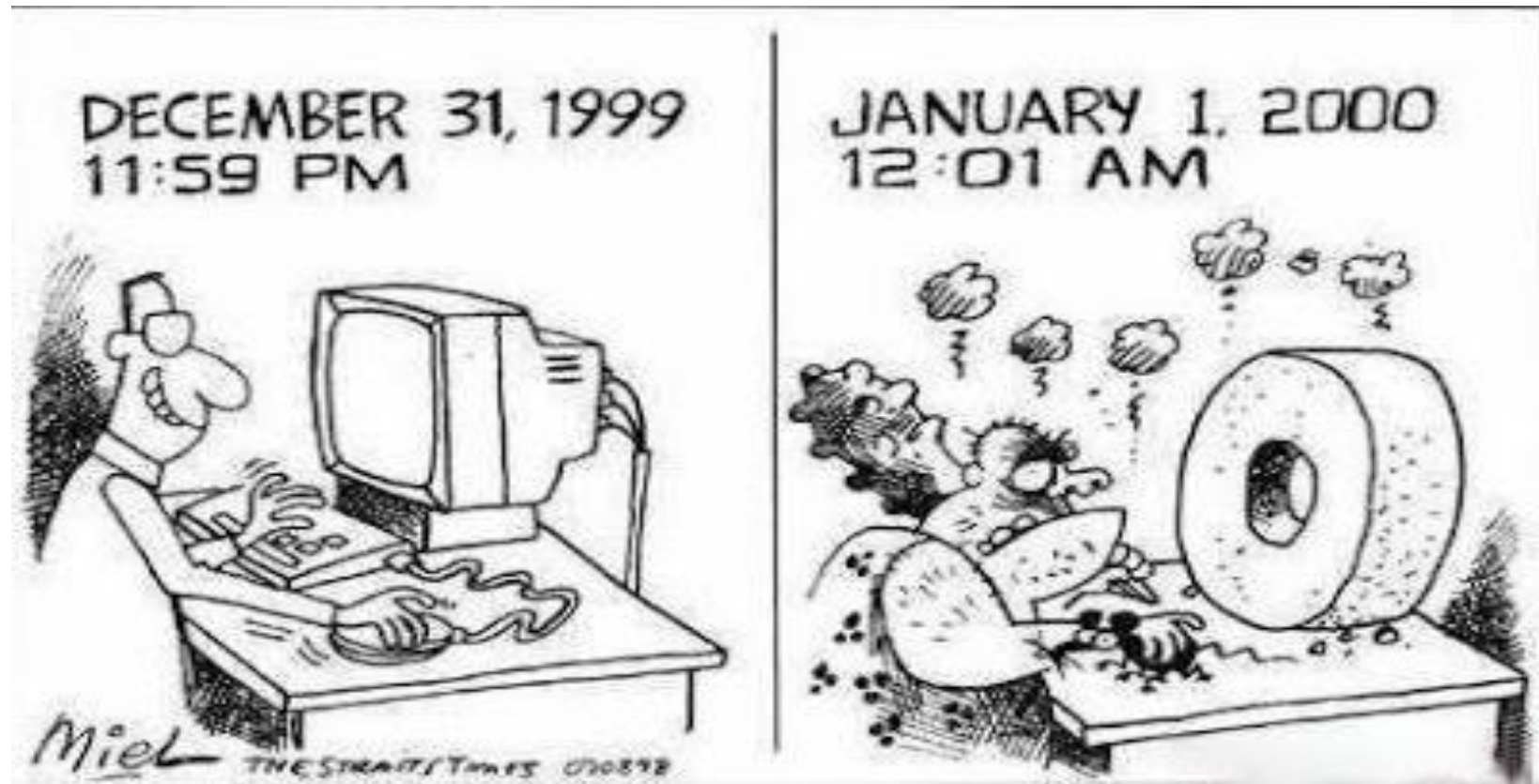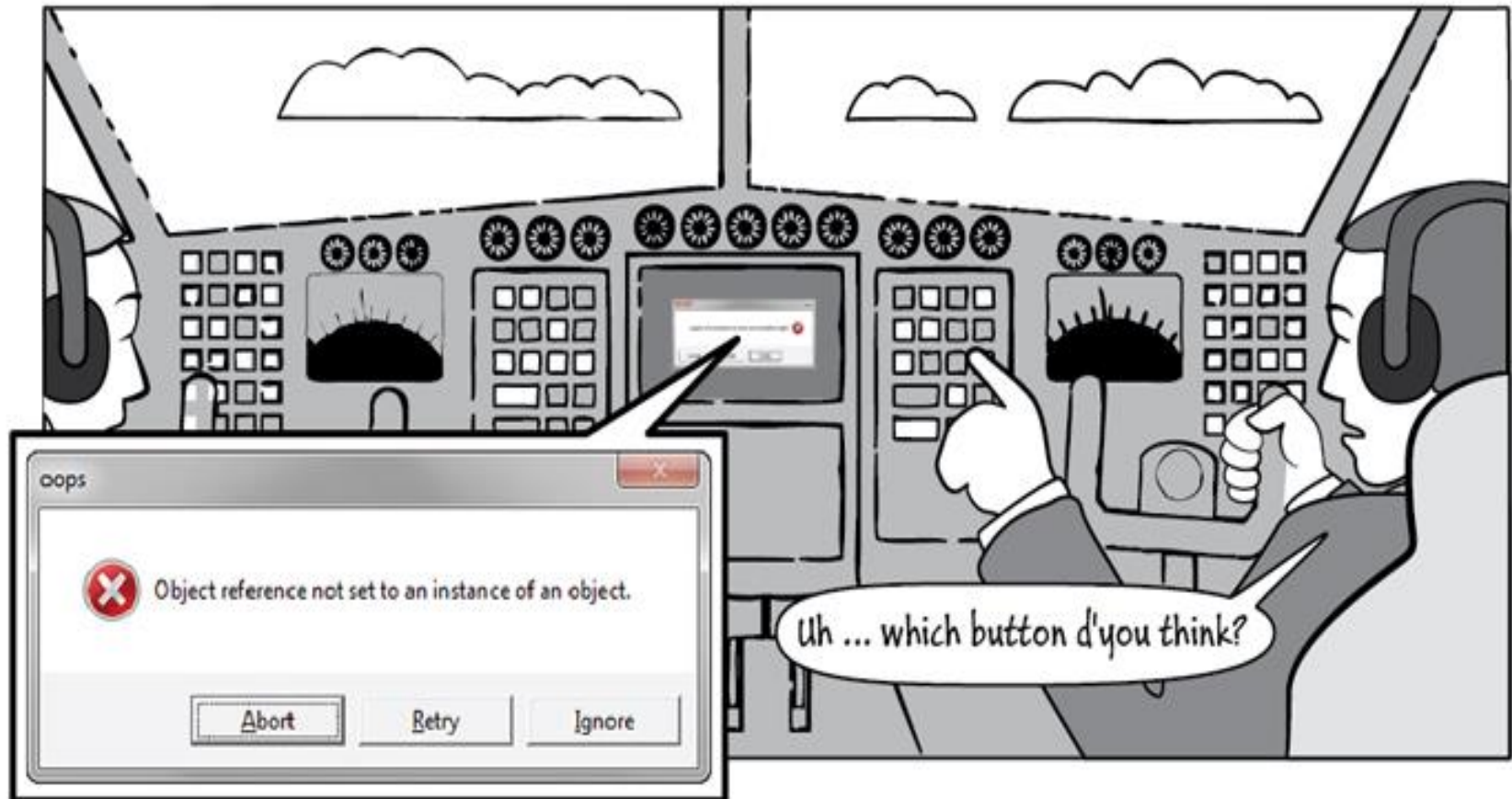
The USA Star- Wars program

LINK

Software Testing Tutorial 1- Why Testing is Important-.mp4

6

TESTING

Reasons for testing

SOFTWARE DEVELOPMENT LIFE CYCLE
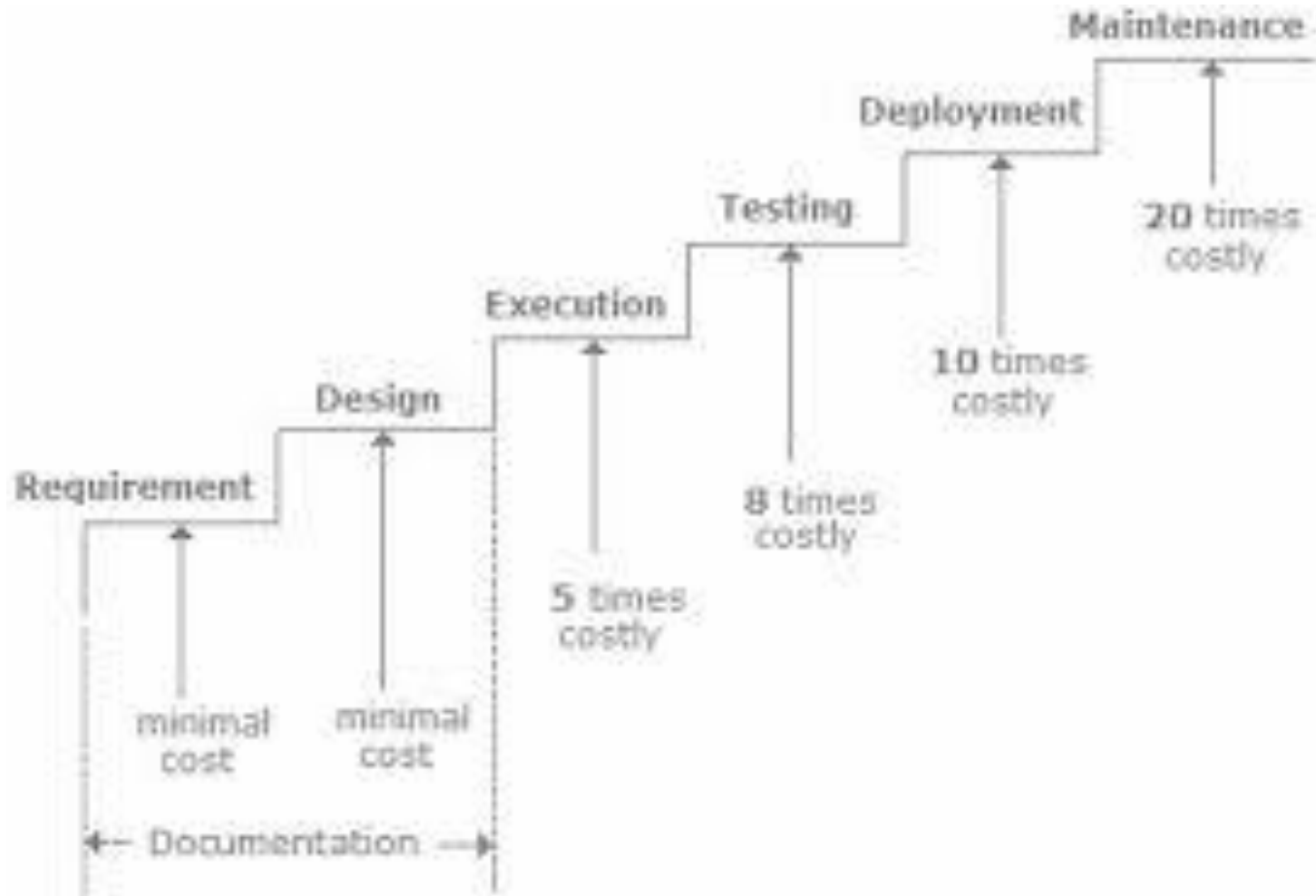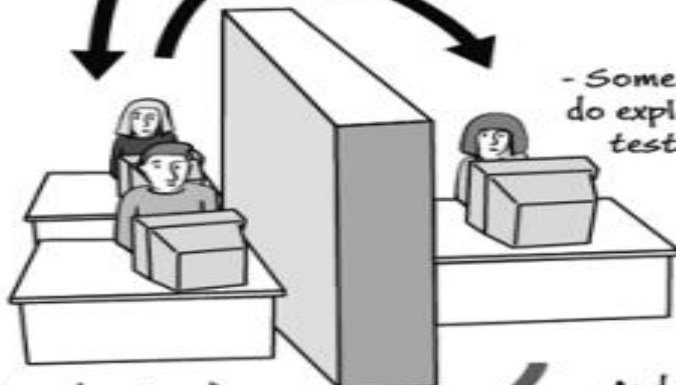
TESTING PROCESS

```
LINE NUMBER      /*SOURCE CODE*/
                 #include<stdio.h>
                 #include<limits.h>
                 #include<conio.h>
1.               void Minimum();
2.               void main()
3.               {
4.                       Minimum();
5.               }
6.               void Minimum()
7.               {
8.                       int array[100];
9.                       int Number;
10.                      int i;
11.                      int tmpData;
12.                      int Minimum=INT_MAX;
13.                      clrscr();
14.                      "printf("Enter the size of the array:");
15.                      scanf("%d",&Number);
16.                      for(i=0;i<Number;i++) {
17.                              printf("Enter A[%d]=",i+1);
18.                              scanf("%d",&tmpData);
19.                              tmpData=(tmpData<0)?-tmpData:tmpData;
20.                              array[i]=tmpData;
21.                      }
22.                      i=1;
23.                      while(i<Number-1) {
24.                              if(Minimum>array[i])
25.                              {
26.                                      Minimum=array[i];
27.                              }
28.                              i++;
29.                      }
30.                      printf("Minimum = %d\n", Minimum);
31.                      getch();
32.              }
```

13

Sample Test cases

Test Case	Size	Inputs — Set of Integers
1.	5	6, 9, 2, 16, 19
2.	7	96, 11, 32, 9, 39, 99, 91
3.	7	31, 36, 42, 16, 65, 76, 81
4.	6	28, 21, 36, 31, 30, 38
5.	6	106, 109, 88, 111, 114, 116
6.	6	61, 69, 99, 31, 21, 69
7.	4	6, 2, 9, 5
8.	4	99, 21, 7, 49

14

SAMPLE TEST CASES

Test Case	Size	Inputs Set of Integers	Expected Output	Observed Output	Match?
1.	5	6, 9, 2, 16, 19	2	2	Yes
2.	7	96, 11, 32, 9, 39, 99, 91	9	9	Yes
3.	7	31, 36, 42, 16, 65, 76, 81	16	16	Yes
4.	6	28, 21, 36, 31, 30, 38	21	21	Yes
5.	6	106, 109, 88, 111, 114, 116	88	88	Yes
6.	6	61, 69, 99, 31, 21, 69	21	21	Yes
7.	4	6, 2, 9, 5	2	2	Yes
8.	4	99, 21, 7, 49	7	7	Yes

TESTING- DEFINITIONS

1. Testing is the process of demonstrating that errors are not present.
2. The purpose of testing is to show that a program performs its intended functions correctly.
3. Testing is the process of establishing confidence that a program does what it is supposed to do.

"Testing is the process of executing a program with the intent of finding faults".

CRITICAL SITUATIONS FOR PROGRAM MINIMUM

i. A very short list with the size of 1,2 or 3 elements.

ii. An empty list

iii. A list where minimum is first or last element.

iv. A list where minimum element is negative.

v. A list where all elements are negative.

vi. A list where some elements are real numbers.

vii. A list where some elements are alphabetic characters.

viii. A list with duplicate elements.

ix. A list where one element has a value greater than maximum permissible limit of an integer.

17

TEST CASES CONSIDERING CRITICAL SITUATIONS

S. No.		Size	Inputs Set of Integers	Expected Output	Observed Output	Match?
Case 1 A very short list with size 1, 2 or 3	A	1	90	90	2147483647	No
	B	2	12, 10	10	2147483647	No
	C	2	10, 12	10	2147483647	No
	D	3	12, 14, 36	12	14	No
	E	3	36, 14, 12	12	14	No
	F	3	14, 12, 36	12	12	Yes
Case 2 An empty list, i.e. of size 0	A	0	–	Error message	2147483647	No
Case 3 A list where the minimum element is the first or last element	A	5	10, 23, 34, 81, 97	10	23	No
	B	5	97, 81, 34, 23, 10	10	23	No
Case 4 A list where the minimum element is negative	A	4	10, –2, 5, 23	–2	2	No
	B	4	5, –25, 20, 36	–25	20	No
Case 5 A list where all elements are negative	A	5	–23, –31, –45, –56, –78	–78	31	No
	B	5	–6, –203, –56, –78, –2	–203	56	No
Case 6 A list where some elements are real numbers	A	5	12, 34.56, 6.9, 62.14, 19	6.9	34 (The program does not take values for index 3,4 and 5)	No
	B	5.4	2, 3, 5, 6, 9	2	858993460 (The program does not take any array value)	No

18

TEST CASES CONSIDERING CRITICAL SITUATIONS(CONTD...)

S. No.		Size	Inputs Set of Integers	Expected Output	Observed Output	Match?
Case 7 A list where some elements are characters	A	5	23, 2l, 26, 6, 9	6	2 (The program does not take any other index value for 3, 4 and 5)	**No**
	B	1l	2, 3, 4, 9, 6, 5, 11, 12, 14, 21, 22	2	2147483647 (Program does not take any other index value)	**No**
Case 8 A list with dupli-cate elements	A	5	3, 4, 6, 9, 6	3	4	**No**
	B	5	13, 6, 6, 9, 15	6	6	**Yes**
Case 9 A list where one element has a value greater than the maximum permissible limit of an integer	A	5	530, 4294967297, 23, 46, 59	23	1	**No**

POSSIBLE REASONS FOR FAILURE

S. No.	Possible Reasons
Case 1 A very short list with size 1, 2 or 3	While finding the minimum, the base value of the index and/or end value of the index of the usable array has not been handled properly (see line numbers 22 and 23).
Case 2 An empty list i.e. of size 0	The program proceeds without checking the size of the array (see line numbers 15 and 16).
Case 3 A list where the minimum element is the first or last element	Same as for Case 1.
Case 4 A list where the minimum element is negative	The program converts all negative integers into positive integers (see line number 19).
Case 5 A list where all elements are negative	Same as for Case 4.

POSSIBLE REASONS FOR FAILURE(CONTD..)

S. No.	Possible Reasons
Case 6 A list where some elements are real numbers	The program uses scanf() function to read the values. The scanf() has unpredictable behaviour for inputs not according to the specified format. (See line numbers 15 and 18).
Case 7 A list where some elements are alphabetic characters	Same as for Case 6.
Case 8 A list with duplicate elements	(a) Same as for Case 1. (b) We are getting the correct result because the minimum value is in the middle of the list and all values are positive.
Case 9 A list with one value greater than the maximum permissible limit of an integer	This is a hardware dependent problem. This is the case of the overflow of maximum permissible value of the integer. In this example, 32 bits integers are used.

REASONS FOR OBSERVED OUTPUT

Cases	Observed Output	Remarks
1 (a)	2147483647	The program has ignored the first and last values of the list. This is
1 (b)	2147483647	the maximum value of a 32 bit integer to which a variable minimum
1 (c)	2147483647	is initialized.
1 (d)	14	The program has ignored the first and last values of the list. The
1 (e)	14	middle value is 14.
1 (f)	12	The program has ignored the first and last value of the list. Fortunately, the middle value is the minimum value and thus the result is correct.
2 (a)	2147483647	The maximum value of a 32 bit integer to which a variable minimum is initialized.
3 (a)	23	The program has ignored the first and last values of the list. The
3 (b)	23	value 23 is the minimum value in the remaining list.
4 (a)	2	The program has ignored the first and last values. It has also
4 (b)	20	converted negative integer(s) to positive integer(s).
5 (a)	31	Same as Case 4.
5 (b)	56	
6 (a)	34	After getting '.' of 34.56, the program was terminated and 34 was displayed. However, the program has also ignored 12, being the first index value.
6 (b)	858993460	Garbage value.
7 (a)	2	After getting 'I' in the second index value '2I', the program terminated abruptly and displayed 2.
7 (b)	2147483647	The input has a non digit value. The program displays the value to which variable 'minimum' is initialized.
8 (a)	4	The program has ignored the first and last index values. 4 is the minimum in the remaining list.
8 (b)	6	Fortunately the result is correct although the first and last index values are ignored.
9 (a)	1	The program displays this value due to the overflow of the 32 bit signed integer data type used in the program.

Modifications in the program 'Minimum'

Reasons for failures:

- The program has ignored first and last values of the list.

- The program proceeds without checking size of an array.

- Program has converted negative values to positive values.

23

NEW TEST CASE

Sr. No.		Inputs		Expected Output	Observed Output	Match?
		Size	Set of Integers			
Case 1						
A very short list with size 1, 2 or 3	A	1	90	90	90	Yes
	B	2	12, 10	10	10	Yes
	C	2	10, 12	10	10	Yes
	D	3	12, 14, 36	12	12	Yes
	E	3	36, 14, 12	12	12	Yes
	F	3	14, 12, 36	12	12	Yes
Case 2						
An empty list, i.e. of size 0	A	0	–	Error message	Error message	Yes
Case 3						
A list where the minimum element is the first or last element	A	5	10, 23, 34, 81, 97	10	10	Yes
	B	5	97, 81, 34, 23, 10	10	10	Yes
Case 4						
A list where the minimum element is negative	A	4	10, –2, 5, 23	–2	–2	Yes
	B	4	5, –25, 20, 36	–25	–25	Yes
Case 5						
A list where all elements are negative	A	5	–23, –31, –45, –56, –78	–78	–78	Yes
	B	5	–6, –203, –56, –78, –2	–203	–203	Yes
Case 6						
A list where some elements are real numbers	A	5	12, 34.56, 6.9, 62.14, 19	6.9	34	No
	B	5.4	2, 3, 5, 6, 9	2	858993460	No
Case 7						
A list where some elements are alphabetic characters	A	5	23, 21, 26, 6, 9	6	2	No
	B	1I	2, 3, 4, 9, 6, 5, 11, 12, 14, 21, 22	2	858993460	No
Case 8						
A list with duplicate elements	A	5	3, 4, 6, 9, 6	3	3	Yes
	B	5	13, 6, 6, 9, 15	6	6	Yes
Case 9						
A list where one element has a value greater than the maximum permissible limit of an integer	A	5	530, 42949672 97, 23, 46, 59	23	1	No

MODIFIED PROGRAM

```
LINE NUMBER      /*SOURCE CODE*/
                 #include<stdio.h>
                 #include<limits.h>
                 #include<conio.h>
1.               void Minimum();
2.               void main()
3.               {
4.                       Minimum();
5.               }
6.               void Minimum()
7.               {
8.                       int array[100];
9.                       int Number;
10.                      int i;
11.                      int tmpData;
12.                      int Minimum=INT_MAX;
13.                      clrscr();
14.                      printf("Enter the size of the array:");
15.                      scanf("%d",&Number);
16.                      if(Number<=0||Number>100) {
17.                              printf("Invalid size specified");
18.                      }
19.                      else {
20.                      printf("Warning: The data entered must be a valid integer and
                         must be between %d to %d. INT_MIN. INT_MAX\n");
21.                      for(i=0;i<Number;i++) {
22.                              printf("Enter A[%d]=",i+1);
23.                              scanf("%d",&tmpData);
24.                              /*tmpData=(tmpData<0)?-tmpData:tmpData;*/
25.                              array[i]=tmpData;
26.                      }
27.                      i=0;
28.                      while(i<=Number-1) {
29.                              if(Minimum>array[i])
30.                              {
31.                                      Minimum=array[i];
32.                              }
33.                              i++;
34.                      }
35.                      printf("Minimum = %d\n", Minimum);
36.                      }
```

PERSONS AND THEIR ROLES DURING SDLC

S. No.	Persons	Roles
1.	Customer	Provides funding, gives requirements, approves changes and some test results.
2.	Project Manager	Plans and manages the project.
3.	Software Developer(s)	Designs, codes and builds the software; participates in source code reviews and testing; fixes bugs, defects and shortcomings.
4.	Testing co-ordinator(s)	Creates test plans and test specifications based on the requirements and functional and technical documents.
5.	Testing person(s)	Executes the tests and documents results.

IMPORTANT TERMINOLOGIES

- Program and Software
- Verification and Validation
- Fault, Errors ,Bug and Failures
- Test cases and test suites.
- Deliverables and Milestones.
- Alpha, Beta and Acceptance testing.
- Quality and Reliability
- Quality assurance and control
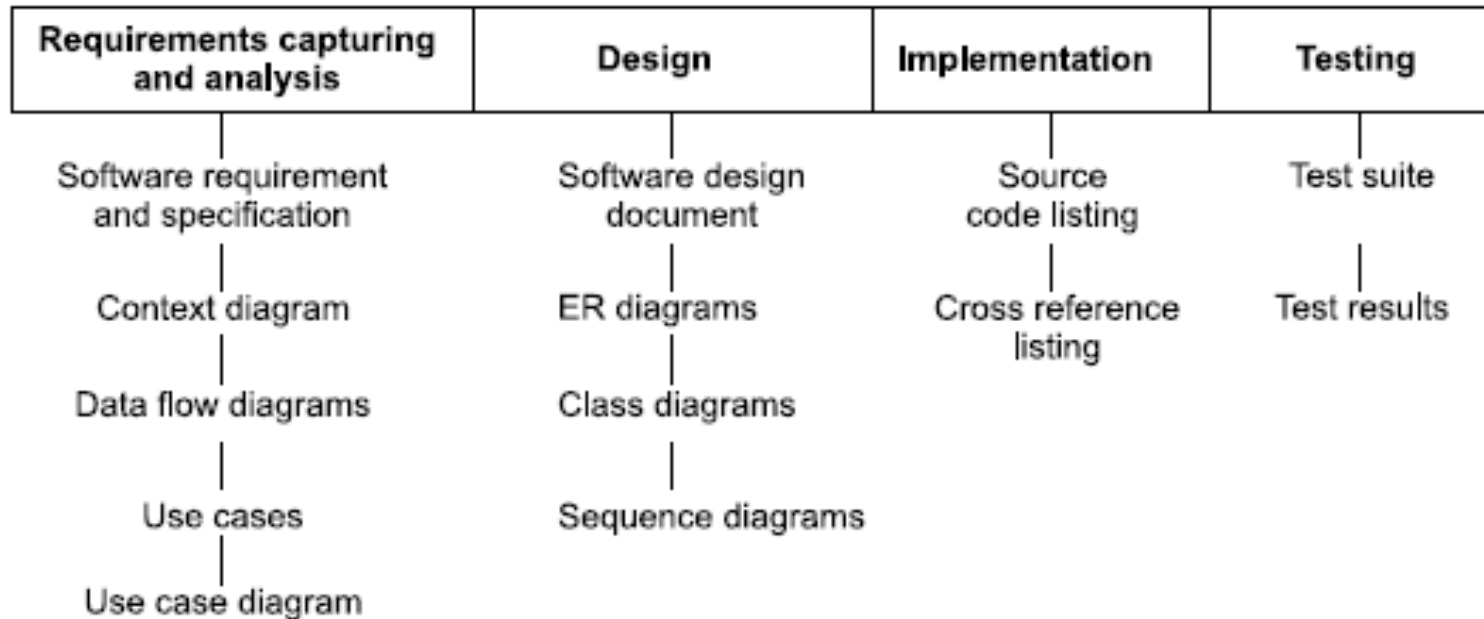- Static and Dynamic testing
- Testing and Debugging.

PROGRAM AND SOFTWARE

❑ Software is the superset of the program(s).

❑ Software consists of one or many program(s), documentation manuals and operating procedure manuals
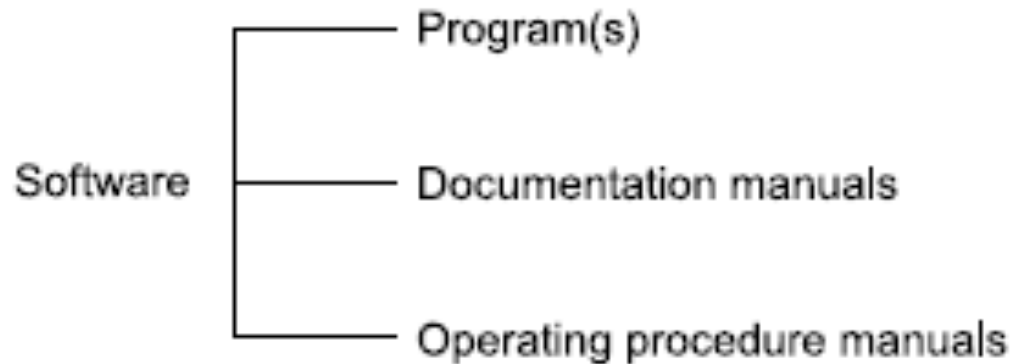
Software ─┬─── Program(s)
          ├─── Documentation manuals
          └─── Operating procedure manuals

Software = Program(s) + Documentation + Operations
                        manuals   procedure manuals

28

DOCUMENTATION MANUALS

Requirements capturing and analysis	Design	Implementation	Testing
Software requirement and specification	Software design document	Source code listing	Test suite
Context diagram	ER diagrams	Cross reference listing	Test results
Data flow diagrams	Class diagrams		
Use cases	Sequence diagrams		
Use case diagram			

Figure 1.7. Documentation manuals

29

VERIFICATION AND VALIDATION

❑ Verification
  ❑ It is the process of evaluating the system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase
  ❑ Verification is related to static testing which is performed manually.
  ❑ Ex: A GUI of login page should contain label and textbox for Username and password and a submit button.

❑ Validation
  ❑ It is the process of evaluating a system or component during or at the end of development process to determine whether it satisfies the specified requirements.
  ❑ Validation is dynamic in nature and requires the execution of the program.
  ❑ For ex : : A GUI of new registration page,Password should contain min of 6 characters with a alphanumeric.
  ❑ In login page, On clicking submit button, home page is displayed.
  ❑ Does software meets user's needs?

  Testing =Verification + Validation

TEST CASE AND TEST SUITE

- A test case consists of inputs given to the program and its expected outputs.
- The set of test cases is called a test suite.

31

DELIVERABLES AND MILESTONES

- Deliverables are generated during various phases of the software development.
  - Software Requirements and Specification document (SRS),
  - Software Design Document (SDD),
  - Installation guide,
  - user reference manual, etc.
- The milestones are the events that are used to ascertain the status of the project.
  - finalization of SRS is a milestone;
  - completion of SDD is another milestone.
- The milestones are essential for monitoring and planning the progress of the software development.

Alpha, Beta and Acceptance testing

- In acceptance testing, software is developed for a specific customer.
- The customer is involved during the acceptance testing process.

- Alpha & Beta Tests are conducted when the software is developed as a product for anonymous customers
- Alpha tests are conducted at the developer's site by the customer.

- Beta tests are conducted by potential customers at their sites.
- The developer is not present during beta testing.

QUALITY AND RELIABILITY

- Software reliability is one of the important factors of software quality.
- Software reliability is defined as "the probability of failure-free operation for a specified time in a specified environment."

- Software quality determines how well the software is designed and how well it conforms to that design.

34

QA and QC

Quality Assurance	Quality Control
Concentrates on process of producing the product.	Concentrates on specific products
Defect prevention oriented	Defect detection and correction oriented
Ex: reviews and audits	Software testing

QA enforces standards and techniques to improve the development process and prevent the previous faults from ever occurring.

Quality control attempts to test the system thoroughly
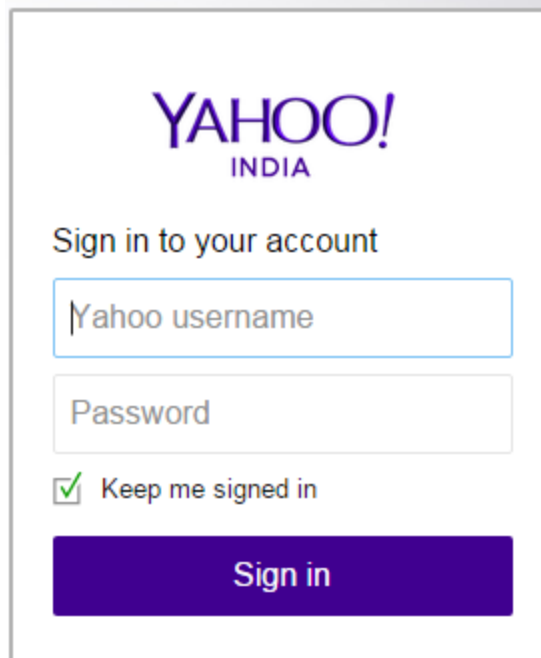
STATIC AND DYNAMIC TESTING

- Static testing refers to testing activities without executing the code.
- Static testing includes verification activities.

- Dynamic testing refers to executing the code.
- Dynamic testing includes validation activities.

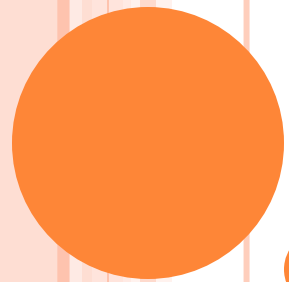- Debugging is the process used to determine the cause of the fault.

36

Fault, Error, Bug and Failure

- A fault is the representation of an error where representation is the mode of expression such as data flow diagrams, ER diagrams, source code, use cases, etc.

- If fault is in the source code, we call it a bug.

- Error /mistake / defect in coding is called a bug

- A failure is the result of execution of a fault and is dynamic in nature.

- A particular fault may cause different failures depending on the inputs to the program.

TEST CASES FOR YAHOO LOGIN

END OF LECTURE