Name: Syeda Nowshin Ibnat

ID: 17183103020

Intake: 39 (1)

1(a)

Soln: Many errors can be found by using a disceliplined thought process without even going near the computer. One such thought process is induction. By starting with the symptoms of the error, possibly in the result of one or more test cases, and looking for relationship among the symptoms, the error is often uncovered. The inductive approach comes from the formulation or a single working hypothesis based on:

→ The data
→ The analysis of existing data
→ The analysis on specially collected data to prove or disprove the working hypothesis.

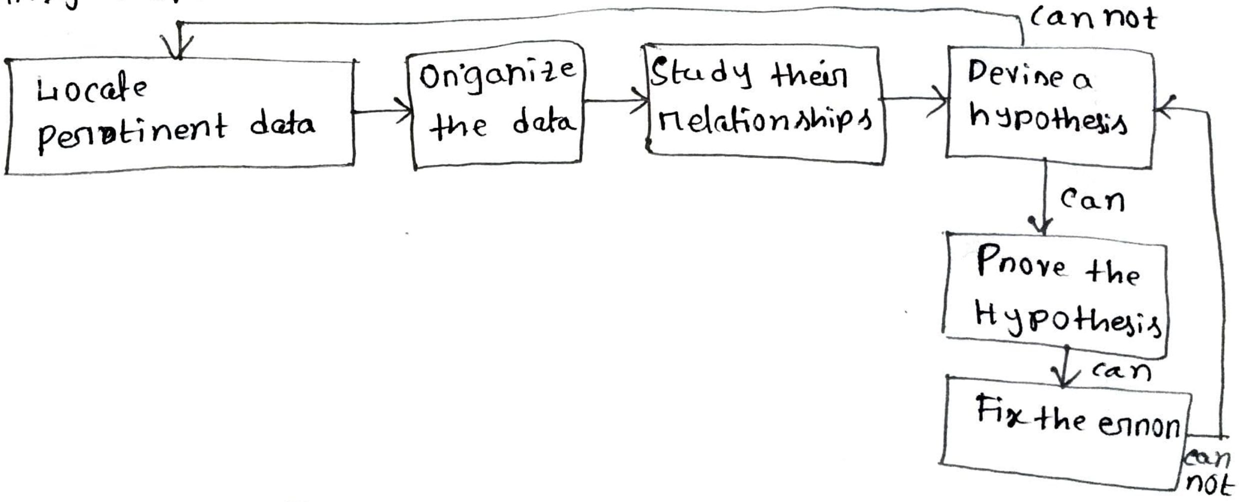


Figure: Inductive Debugging Process

Now Analyzing the case step by step for the e-commerce company:

Locate the pertinent data: This first step is the enumeration of all that is known about what the program did correctly, and what it did incorrectly (i.e. the symptoms that led one to believe that an error exists).

Organize the data: This second step is the structuring of the pertinent data to allow one to observe patterns of particular importance in the search for contradictions. A particularly used organizational technique can be used for this:

?	is	is not
What		
Where		
When		
To What Extent		

Devise a hypothesis: After study the relationship among the clues and devise, using the patterns that might be visible in the structure of the clues, one or more hypothesis about the cause of the error. If they can n't find or devise a theory, then more data are necessary. Here, if multiple theorgies are possible then the most probable one can be selected at first.

Prove the hypothesis: This step is vital to prove the ~~Relat~~ reasonableness of the hypothesis before proceeding. The hypothesis is proved by comparing it to the original clues or data, making sure that this hypothesis completely explains the existence of the clues. If it does not, either the hypothesis is invalid, the hypothesis is incomplete or multiple errors are present.

After proving the hypothesis they have to ~~pro~~ fix the error.

So, Applying these inductive debugging process steps the e-commerce company can easily implement an auto recommendation system for its customer ~~whl~~ while analyzing their shopping pattern.

Ans:

1(b)

Soln: We know,

The overall risk exposure, RE, is determined using the following relationship:

$$RE = P \times C$$

Where, P = The probability of occurrence for a risk.

C = The cost to the project should the risk occure.

Risk identification: Only 50 percent of the software components scheduled for reuse will, infact, be integrated into the application. The remaining functionality will have to be custom developed.

Risk probability: 70%.

Risk impact: 60 reusable software components were planned. If only 50 percent can be used, 20 components would have to be developed from scratch.

Since, the average component is 100 LOC and ~~the~~ local data indicate that the software engineering cost for each LOC is \$20 (my id), the overall cost (impact) to develop the components would be = 20 X 100 X 20

~~= 40000 \$~~

= \$ 40000 (development cost)

The risk exposure,

$$RE = 0.7 \times \$40000$$

$$= \$28000$$

$70\% \leftarrow$ risk probability

$$= \frac{70}{100}$$

$$= 0.7$$

So, the risk exposure is = \$ 28000

Am:

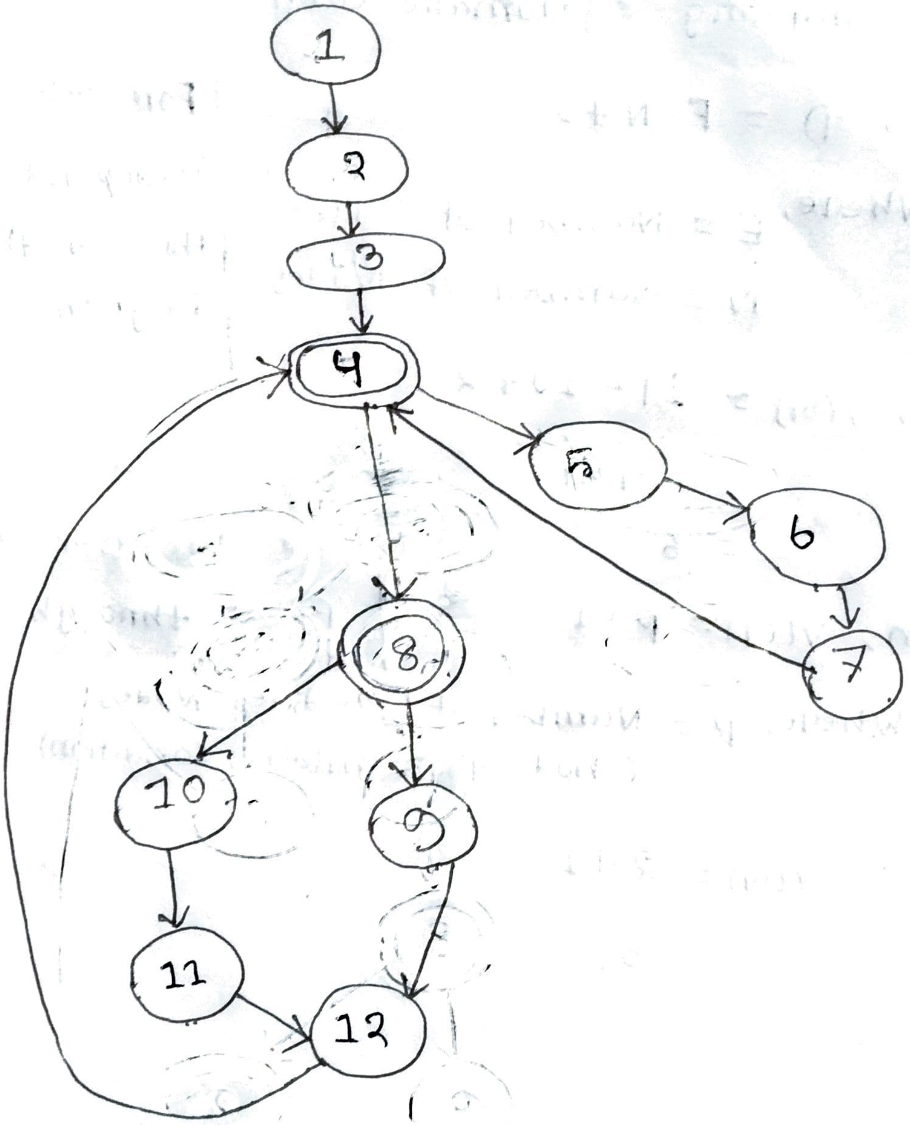
Q (a)

Part-1

Psoln: ~~Pseda~~ pseudo code for identifying palindrome and non-palindrome numbers from it's user input:

```
1. get num.
2. Set ~~rev=0~~ temp= num
3. Set rev= 0
4. While (temp >0)
5. digit = temp % 10
6. rev = (rev* 10)+ digit
7. temp = temp /10
8. ~~8~~ if (num== rev)
9. Printf(" Palindrome");
10. else
11. Printf(" Not palindrome");
12. ~~// End while~~ Printf("\n");
13. // End while
```

BPT Graph



Part-2

Now. computing cyclomatic complexity in below:

☑ $V(G) = E - N + 2$

For the set of independent paths through the graph diagram.

Where, E = Number of edges

N = Number of Nodes

So, $V(G) = 14 - 10 + 2$

$= 4 + 2$

$= 6$

Again, $V(G) = P + 1$

For the set of dependent Paths through the graph diagram.

Where, P = Number of predicted Nodes
(node that contains condition)

So, $V(G) = 2 + 1$

$= 3$

Am:

2(b)

Soln: From 2(a) BPT Graph we can find these possible paths:

Our end node = 12

Since, $V(G) = 3$,
There are thee paths

Path 1: 1 → 2 → 3 → 4 → 8 → 9 → 12

Path 2: 1 → 2 → 3 → 4 → 8 → 10 → 11 → 12

Path 3: 1 → 2 → 3 → 4 → 5 → 6 → 7 → 4 → 8 → 9 → 12

Test cases

#control Structure Testing:

~~1) conditional testing~~

1) Data flow testing - We have to test the paths here. Path-1, Path-2 and Path-3 all of them are connect. But there will be another path which is path-4

Path 4: 1 → 2 → 3 → 4 → 5 → 6 → 7 → 4 → 8 → 10 → 11 → 12.

2) condition testing:- We have to test the logical conditions here. We have 2 conditions (on line-8, 10) from the pseudo code we can see. One of them consist with logical condition.

Loop testing:

From the pseudocode we can see we have a simple while loop. While (temp>0).

Steps to follow:

The following steps should be followed for computing cyclomatic complexity and test cases.

~~1~~ Step 1 - Construction of graph with nodes and edges from the code.

Step 2 - Identification of idependent Paths

Step 3 - cyclomatic complexity calculation

Step 4 - Design test cases

Here, we Fullfilled Step - 1, 2, 3.

Ans:

3(a)

(iv)

Soln: We know, | My ID = 20 ↳ last digit "0"

To compute Function points (FP), the following relationship is used: $FP = \text{count total} \times [0.65 + (0.01 \times \Sigma (F_i))]$

computing Functional points

Informations Domain value	Count		Weighting facton Average	
External Inputes (EIs)	$\boxed{25}$	X	4	= 100
External outputes (EOs)	$\boxed{11}$	X	5	= 55
External queries (EQs)	$\boxed{17}$	X	4	= 68
Internal logical files (ILFs)	$\boxed{5}$	X	10	= 50
~~Internal interfaces~~ External Interface Files (EIFs)	$\boxed{3}$	X	7	= 21
			count total	= 294

Now, we know, if VAF is average then summation of $F_i = 2.5 \times 14 = 35$

So, $FP = ~~35~~ \; 294 \times [0.65 + (0.01 \times 35)]$

$= 294 \times 1$

$= 294$

Ans:

3(b)

(i)

Sol^n: Input range: -120 to $+130$

This input condition specifies a range, one valid and two invalid equivalence classes are defined.

$$\text{Equivalent classes: } \underbrace{\{-120 \,..+130\}\}}_{\text{Valid}},\ \underbrace{\{x < -120\},\{x > +130\}}_{\text{Invalid}} / \{x>130\}$$

(ii)

Input value: $+750$ or we can write ~~is~~ it 750.

This input is a ~~specificie~~ specific value. So, for this we will have one valid and two invalid equivalence classes.

$$\text{Equivalent classes: } \underbrace{\{+750\} \text{ or } \{750\}}_{\text{Valid}},\ \cancel{x}\ \underbrace{\{x<+750\},\{x>+750\}}_{\text{Invalid}}$$

(iii)

Input set: $\{-2,-7,3,5,7,33\}$

Here, given input specifies a member of a set, one valid ~~a~~ and one invalid equivalence class.

$$\text{Equivalent classes: } \underbrace{\{-2,7,3,5,7,33\}}_{\text{Valid}},\ \underbrace{\{\text{any other } x\}}_{\text{Invalid}}$$

(iv)

Input ~~set~~: { F, T, F, F}

~~Equival~~ This given Input specifies • Boolean values, one valid and one invalid.

Equivalent class: {F, T, F, F}, {T, F, T, T} }

↓ ↓

Valid Invalid

(V)

~~Input~~

Input Set: {1,3,5,9,140}

This given input set defines a ~~num~~ member of a set. So, there will be one valid and one invalid equivalence class.

Equivalent class: {1,3,5,9,140}, {any other x}

↓ ↓

Valid Invalid

4(a)

Soln: My ID = 20

So, to make a dynamic Annex Management System and to ~~find~~ deduce the proper designing techniques I have to use → (0) - Content coupling
(2) - Control coupling

(2) Control coupling

Two modules are called control-coupled if one of them decides the function of the other module or changes its flow of execution. Or we can say one module passes an element of control to the other.

Example:

If module p calls module q and q passes back - "I am unable to complete my task", then q is passing data - "I am unable to complete my task; write error ~~message~~ message. ABC 166", then p and q are control coupled.

~~Why~~ ~~But~~ contro coupling is bad ? ~~because~~ - because:

1) Modules are not independent

2) It is associated with modules of logical cohesion.

(0) Content coupling

When a module can directly access or modify or refer to the content of another module, it is called content level coupling. Or we can say one ~~modulbe~~ module directly references contents of the other.

Example:

- Module a modifies statements of module b
- ~~m~~ Module a refers to local data of module b in terms of some numerical displacement within b
- Module a branches into local ~~mo~~ label of module b.

Content coupling is bad because:

From the ~~o~~example any changes on b requires changes on a.

4(b)

Soln: The goal of' the risk management is to identify risks. The risk table is used for risk projection or estimation. A risk table acts as projection tool for project managers to identify and understand different risks.

Risk managing consists with:

1. Risk catagory (Technical issues, business issues, impact risk, stuff size etc.)

2. Estimatio of ~~occur~~ Occurance possibility (Risk possibility)

3. Impact scale of (catastroplic, critical, marginal, ~~me~~ negligible type)

4. Mitigation plan or RMMM plan for specific risk.

The risk Table is given below:- My ID = 20

Risk catagory	Probability level	impact value	Proper comments
(2)- More relese than planned	60%	2	Critical
(0)- Size estimation maybe significantly high	40%	1	Catastrophic

Am: