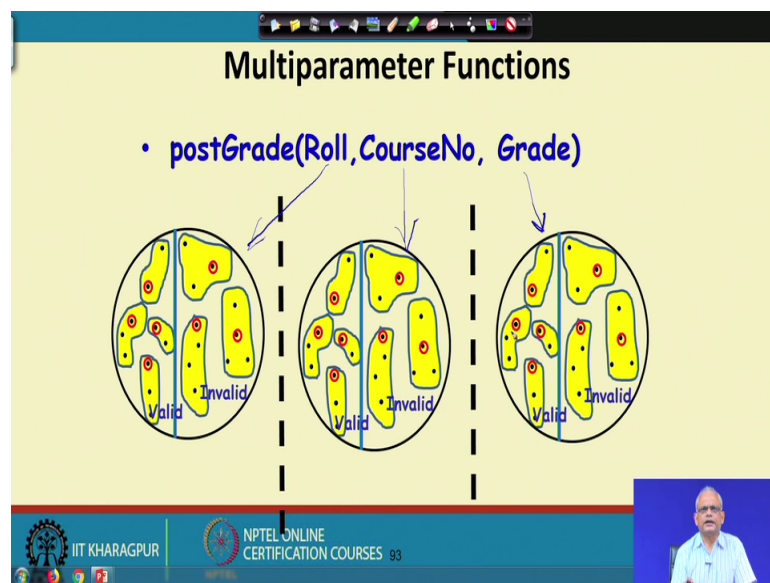


Software Engineering
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 49
Equivalence Class Testing-II

Welcome to this lecture. In the last lecture we were discussing about the Equivalence Class Test cases. We had identified when a function or a unit takes a single parameter, we can examine the input value and then identify the set of equivalence classes, one is the valid set of value, another is a invalid set of value and then for each of the valid and invalid we can identify different sets of equivalence classes, but then we were discussing about multi parameter functions.

(Refer Slide Time: 01:02)



Because it is quite common that a function takes more than one parameter and in that case how do we identify the equivalence class. One way is that we can identify equivalence classes corresponding to each of the parameters here. And then for defining the test case one way is that we define the test cases such that every point here in this set of equivalence classes and a point representative from this set of equivalence classes and a representative from this set of equivalence classes is considered.

The other way is that we have a exhaustive combination of the points from these 3 equivalence classes. So, these are 2 different strategies to define the equivalence class

test cases. We will just look at it further we will develop this idea further in our discussion today.

(Refer Slide Time: 02:14)

```
int Normalization Factor;
postGrade(Roll, CourseNo, Grade)
{ Grade=Grade*NormalizationFactor
-----}
}

Valid Invalid
Valid Invalid
Valid Invalid
Valid Invalid
```

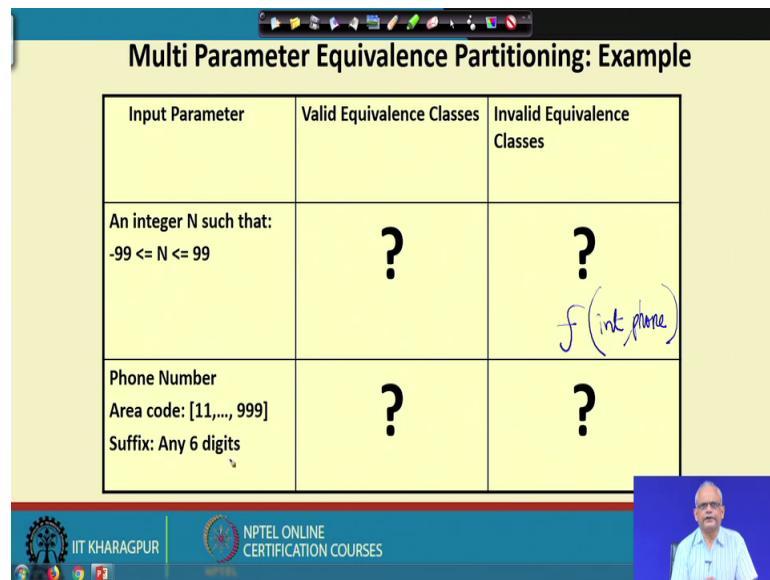
Multiparameter Function Accessing Global Variables

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

But before that we must be aware that not only is that a function like post grade can take the roll number, course number and grade. So, the teacher assigns a grade for a specific course to a student identify by some roll number. There are 3 parameters for this function post grade, but also there can be some global variables for example, normalization factor.

Let us say we have a policy of normalizing the grades and we multiply the grade with a normalization factor. Now we can think of as a additional factor which influences the outcome here. And therefore, we can form equivalence class not only corresponding to the 3 input parameters, but also corresponding to the global variable. Just extension of our discussion of multi parameter function is that a global variable accessed by a function in similar to a parameter to the function. And therefore, we must identify the equivalence classes corresponding to that parameter and then we combine the points from this different equivalence class.

(Refer Slide Time: 04:07)



Input Parameter	Valid Equivalence Classes	Invalid Equivalence Classes
An integer N such that: -99 <= N <= 99	?	? <i>f(int phone)</i>
Phone Number Area code: [11,..., 999] Suffix: Any 6 digits	?	?

Let us do some more practice. Let us say we have a function which takes 2 parameters one is a integer between minus 99 to plus 99. Let me just write the function name as let us say some f name of the function is f. Takes 2 parameters one is a integer parameter and it defines a range between minus 99 and 99. The first parameter takes any value the valid set of values are the minus 99 to plus 99 and the second parameter is a phone number.

The phone number is defined by an area code which is between 1 1 to 9 9 9 9 9 9 and then there is a actual phone number, the area code and the phone number, and the phone number is let us say 6 digit. So, the second parameter is a phone number which is again a integer, but then the first 2 or 3 digits they correspond to the area code and the suffix 6 digits correspond to the phone number. Now how do we identify the equivalence classes for this of course, the first thing is that we identify the valid set of equivalence classes and invalid set of equivalence classes.

For the first parameter the valid set of equivalence is are between any number between minus 99 to plus 99 and then there are 2 invalid equivalence classes, one is less than minus 99 and another is greater than 99, but what about the second parameter. The second parameter again we have a valid equivalence class where we have a 6 digit integer suffix and the first 2 first the prefix that is area code is between 11 and 9 9 9 and

the invalid set of equivalence may arise, because the area code is wrong or may be the suffix is wrong.

(Refer Slide Time: 06:47)

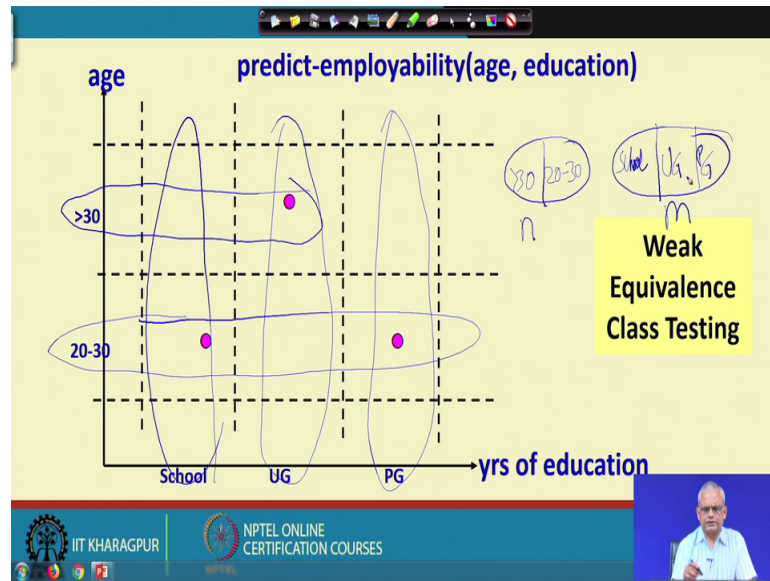
Input	Valid Equivalence Classes	Invalid Equivalence Classes
<p>A integer N such that: $-99 \leq N \leq 99$</p>	<p>$[-99, 99]$</p>	<p>< -99</p> <p>> 99</p> <p>Malformed numbers $\{12-, 1-2-3, \dots\}$</p> <p>Non-numeric strings $\{\text{junk}, 1E2, \\$13\}$</p> <p>Empty value</p>
<p>Phone Number Prefix: $[11, 999]$ Suffix: Any 6 digits</p>	<p>$[11,999][000000,$ $999999]$</p>	<p>Invalid format 5555555, Area code < 11 or > 999 Area code with non-numeric characters</p>

If we write down our idea we will have this one is that we have for the first parameter is a range and then we have one equivalence class which is one is the valid equivalence class which is any value within the range, another is a invalid equivalence class which is less than minus 99, one more invalid class is greater than 99 and then we can have other equivalence classes as well for example, Malformed numbers some are characters and so on. Non-numeric strings empty value etcetera these are the different invalid equivalence classes.

Similarly for the second parameter we have a valid equivalence class where the area code is between 11 to 999 and the phone number is between 000000 to 999999. Now the equivalence invalid equivalence classes and one is the invalid format we do not have the area code and the phone number or we might have the area code which is less than 11 or greater than 999 this are 2 different equivalence classes invalid equivalence classes; area code with nonnumeric characters phone number which is 7 digits phone number which is character and so on.

So, we will have many invalid equivalence classes and one valid equivalence class for each of this.

(Refer Slide Time: 08:40)



And once we identify the equivalence classes for the 2 parameters, then we combine representative values from these two equivalence classes corresponding to the 2 parameter to define our test cases at the simplest. We will discuss about the weak equivalence class testing where let us say we have 2 parameters age and education.

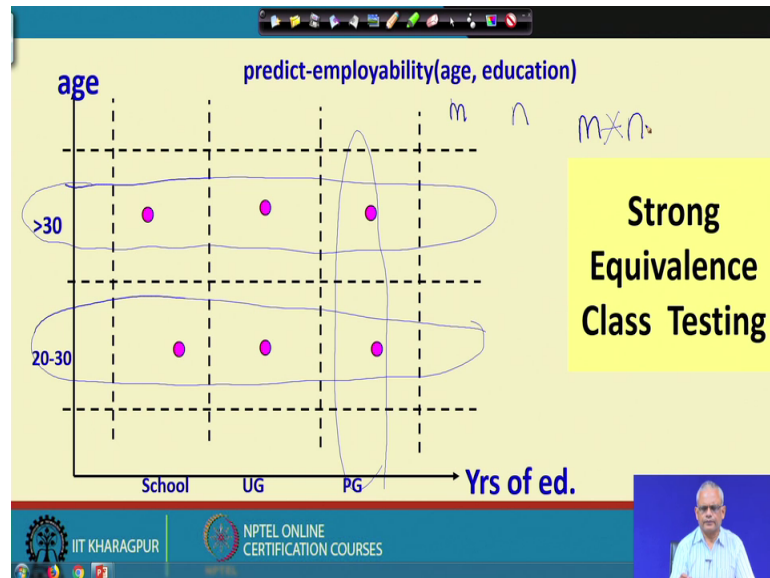
Let us say for the age we identify two equivalence classes, one is age greater than 30 and the second is the age greater between 20 to 30, these are the two equivalence classes. Similarly for the second parameter education let us say we identify 3 equivalence classes school, UG and PG, this is the education. Depending on the years of education we identify whether it is a school education, UG education or a PG education.

Now how do we combine these 2 values from these 2 parameters to form the test cases one way which is called as the weak equivalence class testing is that we see every representative from every equivalence class is represented in the test case. So, we see that every equivalence class from the first parameter is covered. And similarly every representative of the equivalence classes of the second parameter are also covered.

So, if we have let us say n equivalence classes for the first parameter and let us say m equivalence classes of the second parameter then how many test cases we need to be able to do a weak equivalence class testing for this function, can think over it but then. Let me just tell you that we will just take maximum of n and m . And that will form the number

of test cases required to carry out weak equivalence class testing, which will have every representative from every equivalence class to be part of a test case.

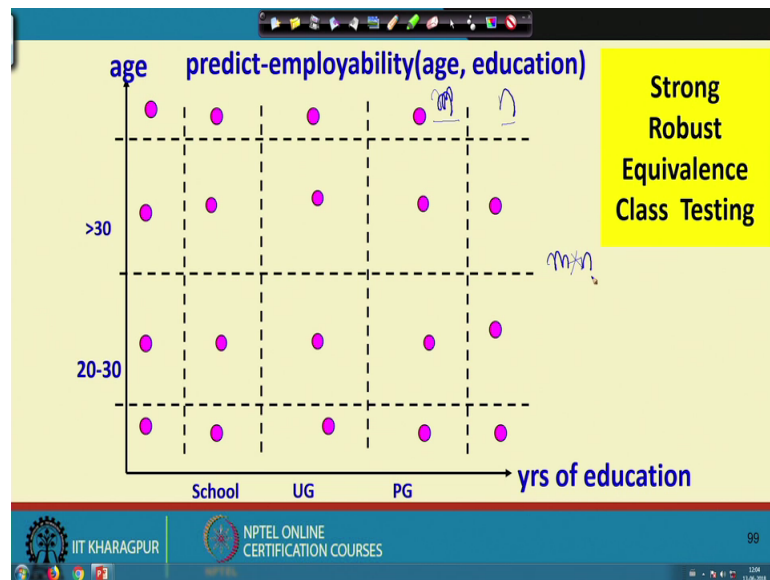
(Refer Slide Time: 11:55)



The other way we can combine the representative values from different equivalence classes is called as the strong equivalence class testing. Here every value of one equivalence class of one parameter is combined with every equivalence class representative every equivalence class of the other parameter. So, for the first parameter here of age, we have a one representative from school, UG, and PG. Similarly for the second equivalence class of the age we again combine with the 3 equivalence classes of the second parameter which is the year of education.

Or, in other words for every value of the second parameter we combine with every other value of the first parameter and this is called a strong equivalence class testing. So, the question here is that, how many test cases will be needed if we have 2 parameters and let us say we have m equivalence classes for the first parameter and n equivalence classes for the second parameter. Since for every equivalence class here we must combine with all equivalence class of the second parameter we need m into n test cases.

(Refer Slide Time: 13:47)



Based on the equivalence classes, we can also combine them in another way which is called as strong robust equivalence class testing. If you look at the 2 types of testing we considered earlier we did not consider the invalid set of equivalence classes we just considered the valid equivalence classes.

But if we consider the invalid set of equivalence classes also, then we call it as a robust testing the idea is simple just a small extension of the strong equivalence class testing here we consider the invalid values as well and this forms our set of test cases. If we if the valid plus invalid we have m equivalence classes for first parameter n for the second parameter. So, these are the some of the valid equivalence classes and invalid equivalence classes then the number of test cases needed for equivalence class testing will be m into n .

(Refer Slide Time: 15:04)

• Design Equivalence class test cases: **compute-interest(days)**

• A bank pays different rates of interest on a deposit depending on the deposit period.

Quiz 1

- 3% for deposit up to 15 days
- 4% for deposit over 15 days and up to 180 days
- 6% for deposit over 180 days upto 1 year
- 7% for deposit over 1 year but less than 3 years
- 8% for deposit 3 years and above

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 100

Let us see let us try to solve some examples, let us say for a bank application we have a function the name of the function is compute interest. And the parameter is days which is a integer and the function description is that it should compute 3 percent interest for deposits less than 15 days, 4 percent for deposit between 15 days to 180 days, 6 percent for 180 days up to 1 year and 7 percent for between 1 to 3 year and 8 percent for 3 years and above.

So, what will be the equivalence classes here, this is a rather straight forward example we have just one parameter and we examined the output and each of these correspond to a scenario and then we consider one equivalence class corresponding to deposit between 0 to 15 days, 15 to 180 days and so on. And the invalid set of inputs which is an invalid day negative day and so on.

(Refer Slide Time: 16:41)

- Design Equivalence class test cases: **compute-interest(principal, days)**
- For deposits of less than or equal to Rs. 1 Lakh, rate of interest:
 - 6% for deposit upto 1 year
 - 7% for deposit over 1 year but less than 3 years
 - 8% for deposit 3 years and above
- For deposits of more than Rs. 1 Lakh, rate of interest:
 - 7% for deposit upto 1 year
 - 8% for deposit over 1 year but less than 3 years
 - 9% for deposit 3 years and above

Now, let us try slightly more complex problem where we have the compute interest, but then it takes 2 parameters principle amount deposited. And the days for which the deposit is made and the function description says that, if the deposit amount that is the principle is more than 1 lakh then the rate of interest is 7 percent, 8 percent, 9 percent, but if the deposit is less than 1 lakh then it is 6 percent, 7 percent, 8 percent.

So, we see here that for the 2 parameters, we have 2 equivalence classes one is more than 1 lakh, less than 1 lakh and the days is 1 year, 3 year and more than 3 years. So, we can have either a weak equivalence class testing or a strong equivalence class testing as required.

(Refer Slide Time: 17:56)

Quiz 3

- Design equivalence class test cases.
 - Consider a program that takes 2 strings of maximum length 20 and 5 characters respectively
 - Checks if the second is a substring of the first
 - **substr(s1,s2):**

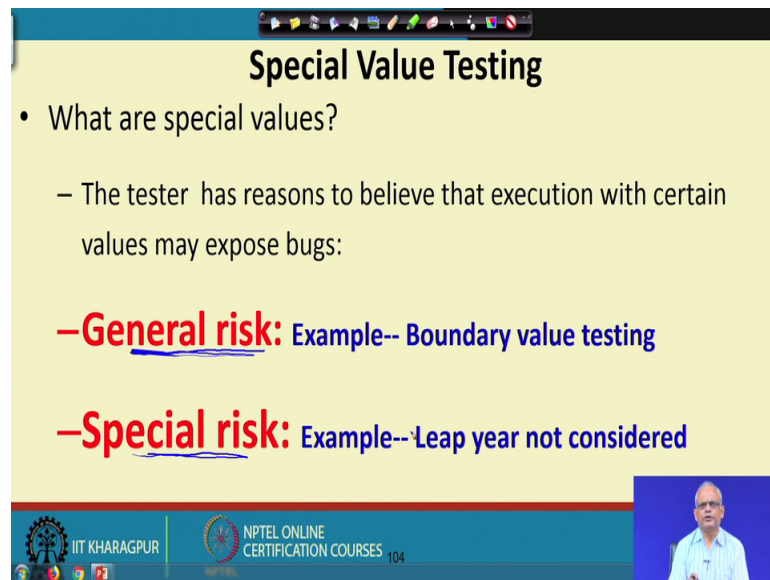
Handwritten diagram: The word "Data" is written in blue. Below it, "s1" and "s2" are written. Arrows point from "s1" to a circle labeled "Valid" and from "s2" to a circle labeled "Invalid".

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 102

This is another problem, let us say we have a function called as substring takes 2 parameters s 1 and s 2, s1 is a string whose length is between 20 is up to 20 characters and the s 2 is a string whose length is less than 5 characters and the function takes whether the second string is a substring of the first string. Again we can form 2 equivalence class hierarchy one is based on the string which is less than invalid set is more than 20 or it can be a control character and so on.

And then the valid set of strings for s 1 valid and invalid set of s 2 and then we would have another equivalence class hierarchy which is whether it is a substring of this or not a substring of this. If we represent here the data input data for s 1 we can have invalid and a valid set of equivalence class similarly for s 2 we will have valid and invalid. And also by looking at the s one and s 2 for valid s 1 and s 2 we can have another hierarchy whether it is a substring or not a substring.

(Refer Slide Time: 20:06)



Special Value Testing

- What are special values?
 - The tester has reasons to believe that execution with certain values may expose bugs:
 - **General risk:** Example-- Boundary value testing
 - **Special risk:** Example-- Leap year not considered

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 104

Now let us start discussing about another type of black box testing called as special value testing. In the equivalence class testing we divided the input data domain into equivalence classes such that every point belonging to one equivalence class resulted in similar behavior, but here there are some values which are suspect.

The tester has reasons to believe that execution of the program with certain specific values may expose the bugs there are 2 types of risky values, that tester can identify one is called as the general risk. The general risk this is identified by examining the equivalence classes and the boundary values of the equivalence classes in general are suspect.

But then there are special values which are of risk which may not be boundary for example, the tester might think that for this function which takes a year and produces the number of days in the year is the leap year considered by the programmer or not. So, this tester by looking at the functionality he guesses some values which might be proven to error and those are called as the special risk values. In addition to equivalence class testing we need to do special value testing and one type of special value testing is the general risk an example is the boundary value testing and also the special risk data values should be used to define test cases.

(Refer Slide Time: 22:16)

The slide features a yellow background with a blue header and footer. A yellow box on the right contains the title 'Boundary Value Analysis'. A diagram shows a horizontal line with a blue segment in the middle and red segments at the ends. The number '1' is circled at the left end of the blue segment, and '100' is circled at the right end. The text on the slide discusses programming errors at these boundaries.

- Some typical programming errors occur:
 - At boundaries of equivalence classes
 - Might be purely due to psychological factors.**
- Programmers often commit mistakes in the:
 - Special processing at the boundaries of equivalence classes.**

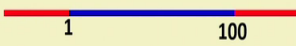
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 105

Now, let us look at the boundary value analysis, this is a general value general risk whenever we identify equivalence classes. The programmers are very proven to commit errors at the boundary of the equivalence classes, here there is one valid set of equivalence class valid set of values which it constitute an equivalence class, one invalid set of value constituting an in invalid equivalence class and another invalid equivalence class.

The programmers are likely to commit errors at the boundary here for example, here whether one is included as in the valid or invalid they might not have taken care of that sufficiently well. And therefore, mistakes may occur here at the boundaries. And therefore, we must include the data values at the boundary as test cases and those are called as boundary value test cases.

(Refer Slide Time: 23:38)

Boundary Value Analysis

- Programmers may improperly use $<$ instead of $<=$
- Boundary value analysis: 
 - Select test cases at the boundaries of different equivalence classes.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 106

The main reason here is that programmer sometimes confuse and instead of less they might use a less than equal to symbol and so on and that is the reason we need to select test cases at the boundaries of different equivalence classes. If boundaries exist we had seen that if it is a set input domain is a set of values then there is no boundary, but if it is a range of values then there are boundaries between different equivalence classes.

(Refer Slide Time: 24:19)

Boundary Value Analysis: Guidelines

- If an input is a range, bounded by values a and b :
 - Test cases should be designed with value a and b , just above and below a and b .
- **Example 1:** Integer D with input range $[-3, 10]$,
 - test values: $-3, 10, 11, -2, 0$
- **Example 2:** Input in the range: $[3, 102]$
 - test values: $3, 102, -1, 200, 5$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 107

Now, let us see if we have a input is a range, it is a range and the boundaries are, a and b , then we need to select not only a , but we need to select one that is beyond a just to check

whether the programmer has made a mistake and even the one which is should be a invalid or a different equivalence class is also considered as part of this equivalence class.

Similarly, we might take a value just inside and similarly for b 1 on b another slightly inside and another just outside. So, if the range is defined as any value between minus 3 to 10 we can take minus 3, we can take 10, these are the two boundaries one slightly away outside the boundary which is minus 2, 11 and one which is just inside the boundary this actually can be the equivalence class test.

So, we need to define 5 values for a range similarly the input is a range it is a 3 to 102 we will define one at boundary 3, one is 102 included the boundary in the test case, one beyond the boundary on the lower side which is minus 1, another beyond the boundary on the higher side which is 200 and one in between which is 5.

(Refer Slide Time: 26:21)

Boundary Value Testing Example

- Process employment applications based on a person's age.

0-16	Do not hire
16-18	May hire on part time basis
18-55	May hire full time
55-99	Do not hire

- Notice the problem at the boundaries.
 - Age "16" is included in two different equivalence classes (as are 18 and 55).

The slide includes a navigation bar at the top, the IIT KHARAGPUR logo, the NPTEL ONLINE CERTIFICATION COURSES logo, and a small video inset of a speaker in the bottom right corner.

Let us see for some specific examples, how do we define the boundary value test cases let us say for some problem where the function is based on the age it recommends whether can be hired or not hired checks if the age is between 0 to 16 it displays do not hire, if it is between 16 to 18 displays can hire on part time basis, if it is between 18 to 55 displays may hire full time and between 55 to 99 do not hire.

So, there are 4 equivalence classes here, corresponding to this 4 different scenarios and the boundaries are at 16 18 and 55, but then we must realize that this description of the function itself is a bit problematic because in 16 considered as do not hire or is it may hire on part time basis similarly if it is exactly 18, what should be the output similarly, 55.

So, even while describing the problem we can commit mistake at the boundary as exemplified here that the boundary values we have not taken care sufficiently it is the specification of the problem itself is a defective and not clearly identified the boundary should belong to which equivalence class. And of course, a same thing will get reflected in the program and therefore, boundary value testing is very important.

We will stop here, and we will continue in the next class.

Thank you.