

Programming, Data Structures and Algorithms
Prof. Shankar Balachandran
Department of Computer Science and Engineering
Indian Institute Technology, Madras

Module - 01
Lecture - 08
Solution: Digital Root Programming Assignment

Good morning all. So, I want to do a small problem session. So, there was this problem called digital root of an integer that was given as part of the assignment, and I thought I will show you how to solve it live. So, we will talk about the solution as we do it on paper and pencil, and then we will see how to translate that to a program. So, I am going to take five examples here namely, 1233, 1234 and so on. And I will show you what really happens in each one of them.

(Refer Slide Time: 00:36)

The screenshot shows a Windows Journal window with the title "Digital Root of an Integer" and a date of "17-03-2014". The content is handwritten in red ink and shows the following calculations:

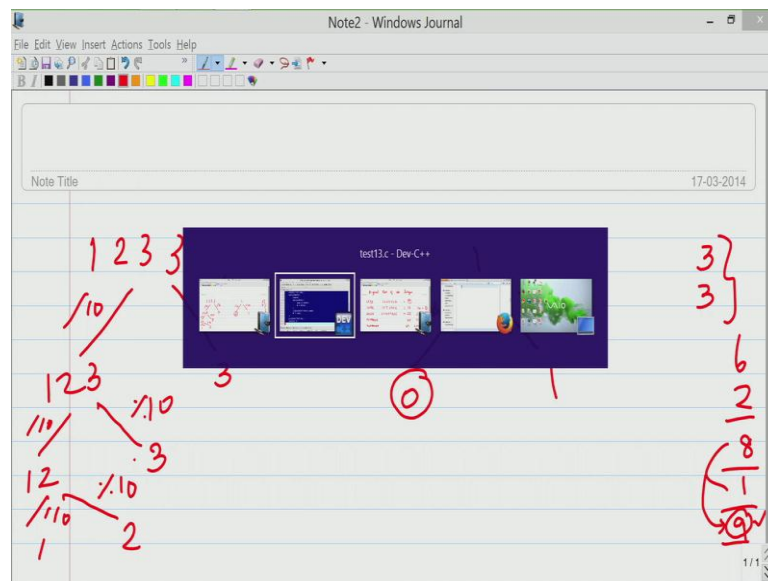
Number	Sum of Digits	Result	Final Digit
1233	$1+2+3+3$	$= 9$	9
1234	$1+2+3+4$	$= 10$	$1+0 = 1$
65536	$6+5+5+3+6$	$= 25$	$2+5 = 7$
9199999		55	$5+5 = 10$
96999999		69	$6+9 = 15$

So, the digital root is defined as sum of all the digits in a number. So, for 1233, it is 1 plus 2 plus 3 plus 3. So, in this case, the submission is 9. And once you hit a single digital number, we stop there. So, 9 is a single digit number; we stop there; however, for 1234, it is 1 plus 2 plus 3 plus 4. So, in this case, the result would be a 10; but we do not stop it there, because 10 is a two digit number. We take 1 and 0 and add it together; we get 1. So, 1 is the final answer for this. Let us take this example 65536; so 6 plus 5 plus 5

plus 3 plus 6. So, that is 6 plus 10 – 16, 19, 25. So, this is 25. So, 2 plus 5 is 7; that is already a single digit number; we can stop. Then, I have taken this slightly larger example – 9 plus 1 plus five 9's. So, that is six 9's, which is 54 plus 1; which is 55. But now, you add 5 and 5 together; you get a 10. We cannot stop here. In the previous cases, you could have stopped slightly earlier; but now, we cannot stop.

Now, add 1 plus 0 together; the result is a 1. And this last example, where we have 9-6 followed by six 9's. And that would be seven 9's, which is 63 plus 6, which is 69. Now, you add 6 plus 9 together; that gives you 15; and 1 plus 5 together; which is 6. So, when we hit a single digit number, we can stop. So, this is the definition of a digital root. And what I have done is it looks like I can see 1233 here and I can extract the digits. But you cannot do this in a program like this. So, instead, what I am going to do is I am going to show you how to write a program for this.

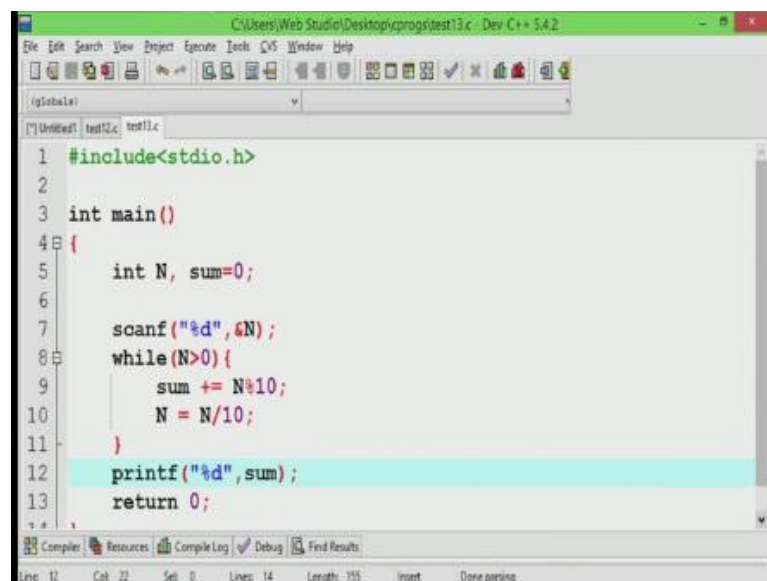
(Refer Slide Time: 02:32)



So, let us look at 1233. What do we need for 1233? I want to extract one digit at a time. And to extract one digit, I am going to use two operators namely, the div operator or slash and mod operator or percentage. So, if I do a percentage 10 of this, I get 3, which is the last digit; but if I do a slash 10 of that or divide by 10; so that is an integer; 10 is an integer. If I divide, I will get 123. So, this gives me one digit. So, let me write it down

here. Then, I take 123; I do a mod 10; I get 3. So, again I need to note it down. And 123 div 10 would be 12. So, I can add this up. The result is now 6. Then, I take 12 itself; do a mod 10; that gives me 2. So, I can add 6 and 2; that is 8. And 12 divided by 10 is 1. And finally, if I take 1 modulo 10, that is 1; and 1 div 10; this is 0. So, 1 modulo 10 is a 1 if I add it together. So, 8 plus 1 is 9. So, at this point, it is a single digit number; and we have destroyed the value of n. So, we have extracted all the digits of the number, that is, input. So, we can stop and print this result. So, in this case, we have got slightly lucky, because we went over one iteration of all the numbers and we ended up with a single digit number itself. So, this may not be the case. But let us first write a program to add up all the digits of a number and then we will worry about if the number is greater than 9 or not. So, let us write a small program for that.

(Refer Slide Time: 04:22)



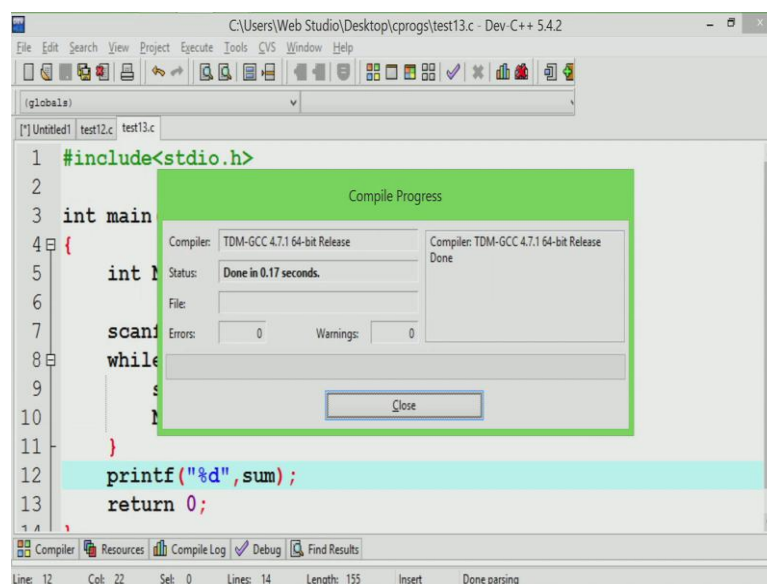
```
1 #include<stdio.h>
2
3 int main()
4 {
5     int N, sum=0;
6
7     scanf("%d",&N);
8     while(N>0){
9         sum += N%10;
10        N = N/10;
11    }
12    printf("%d",sum);
13    return 0;
14 }
```

So, I am going to write... I am going to scan a number called n. But before that, we need the basic template. So, that is the basic template. And what am I going to do? I am going to have an integer n. But since I am also adding digits, we need two variables. So, one is going to accumulate the sum and another is going to be n itself. So, I need a sum. And it is always a good practice to initialize things that are being summed up. So, I will do it right here; sum equals to 0 is initialized. And I am going to scan the number from the user. So, I am scanning the number. Notice that, I did not prompt the user to say like

enter the number or anything; I am just scanning because that is what the portal takes. So, I am scanning the number directly.

And what am I going to do? I am going to write a loop now; it says while n is greater than 0; because once you hit 0, we know that we have actually extracted all the digits. So, what I am going to do is – I am going to add sum plus equals n mod 10. So, that extracts the last digit of n. But I also want to destroy n by a factor of 10 every round. So, I do n equals n by 10. And what do you get here? So, at the end of this, the sum will be the summation of everything that we have seen so far. So, this is not the final solution to the problem. So, if I put sum here, I will see that.

(Refer Slide Time: 06:04)



```
1 #include<stdio.h>
2
3 int main
4 {
5     int n;
6     scanf("%d", &n);
7     while(n > 0)
8     {
9         int sum = 0;
10        sum = sum + n % 10;
11        n = n / 10;
12    }
13    printf("%d", sum);
14    return 0;
```

Compile Progress

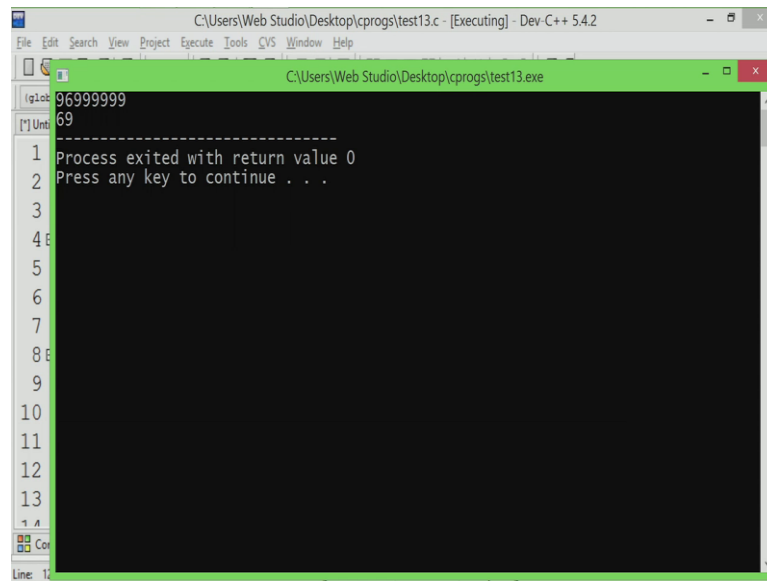
Compiler: TDM-GCC 4.7.1 64-bit Release	Compiler: TDM-GCC 4.7.1 64-bit Release
Status: Done in 0.17 seconds.	Done
File:	
Errors: 0	Warnings: 0

Close

Line: 12 Col: 22 Sel: 0 Lines: 14 Length: 155 Insert Done parsing

So, let me save this; I am going to save it as a program. I have saved this and I will compile it. So, thankfully, there were no compilation errors.

(Refer Slide Time: 06:14)

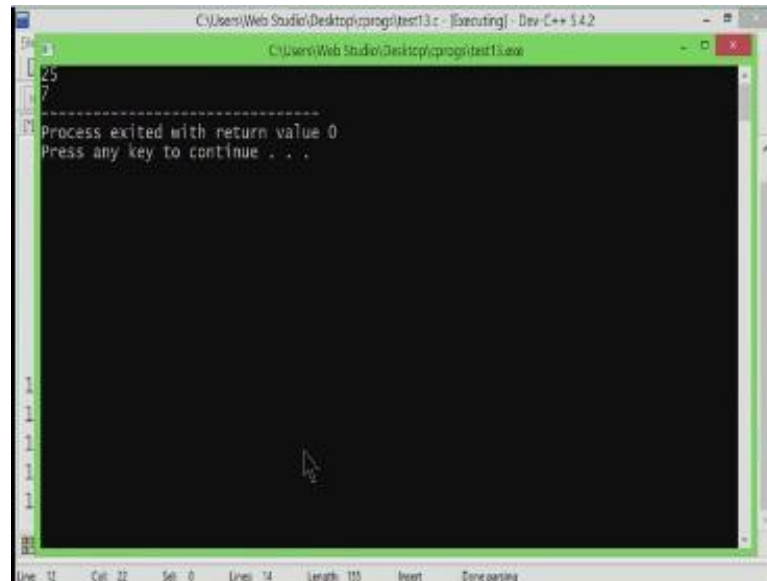


```
C:\Users\Web Studio\Desktop\cprogs\test13.c - [Executing] - Dev-C++ 5.4.2
96999999
69
-----
1 Process exited with return value 0
2 Press any key to continue . . .
3
4
5
6
7
8
9
10
11
12
13
```

I will run it. And I am going to try these examples. So, let us start with 1233 as an example. So, let me try it alone – 1233. And 1233 gives me 9 as expected; so which is good. So, I am going to try it on all the examples that I have just to ensure that, this first step is correct; because if this step is wrong, everything else is going to be wrong. So, 1234 – if add up; it is 10. Then, 65536 should add up to 25, which is good. Then, 911234 – five 9's should add up to 55, which is also correct. And finally, 96 followed by six 9's should add up to 69. So, we have... So far, we seem to have taken all the digits and added them up. And we got a multi-digit number. but however, in this case, except for 1233, all the other things gave two digit numbers.

And we may have to do this process once more. So, the program that we have is currently not sufficient. Instead, we need to take this new sum as n and repeat the process. So, I will just re-iterate what I said. So, at this point, in line number 12, we know that, this is a program, which can take the digits of a number and sum it up. But then this may result in a multi-digit number by itself. So, if I give you a sufficiently large number, it may not even be a two digit number; it may become a three digit number. So, however, the result is stored in a sum. But if I run the same program on that number once more, I will get the result; correct?

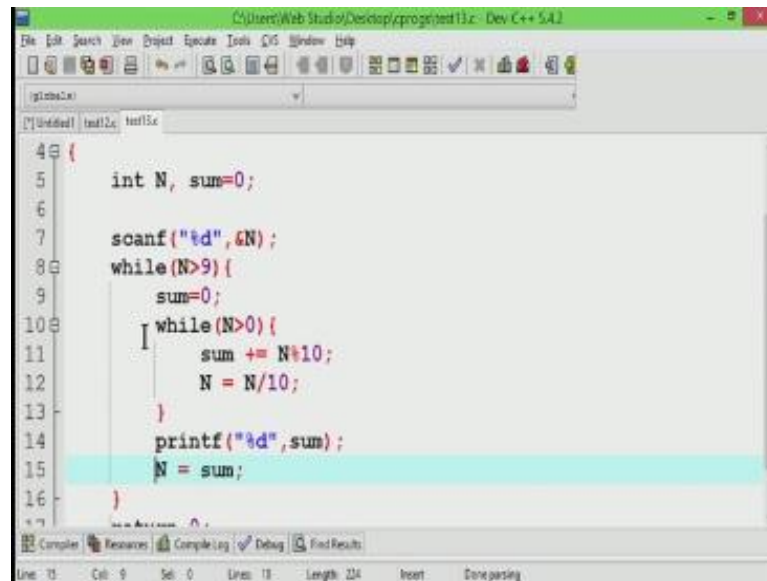
(Refer Slide Time: 08:02)



```
C:\Users\Web Studio\Desktop\prog1\test13.c - [Executing] - Dev C++ 14.2
C:\Users\Web Studio\Desktop\prog1\test13.exe
25
7
-----
Process exited with return value 0.
Press any key to continue . . .
```

So, let us say I put in 65536. So, I got 25. I can always ask the user – run the program once more with 25 in it; you get 7. At this point, you stop. But that is not a good way to write programs. In this case, we are asked to write a program, which will do this whole process without having the user to run the program multiple times. So, the key thing is at this point, this summation – the sum has the value that you need. And what I am going to do is I am going to take N as sum. And let us say I have a mechanism by which I take this point from line number 13; I need to go back to line number 8; and I iterate this whole process once more. If I am capable of doing that; then, this would solve the problem.

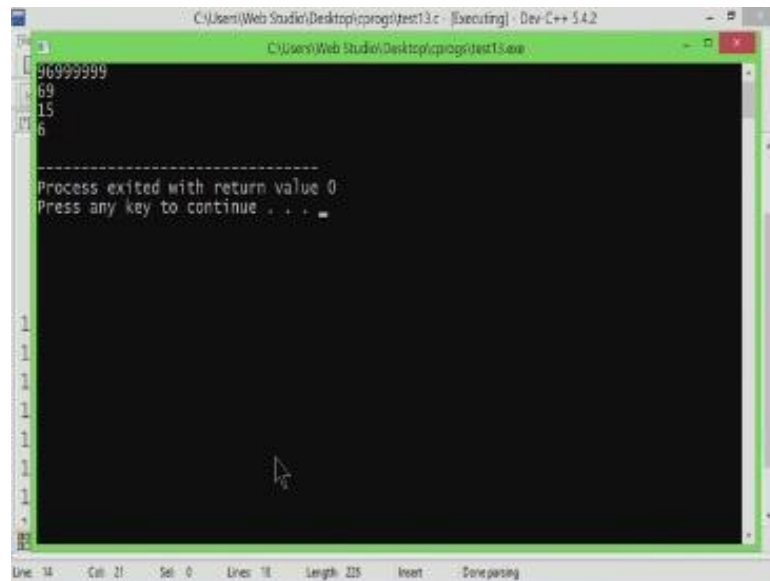
(Refer Slide Time: 08:52)



```
4 {  
5     int N, sum=0;  
6  
7     scanf("%d",&N);  
8     while(N>9){  
9         sum=0;  
10        while(N>0){  
11            sum += N%10;  
12            N = N/10;  
13        }  
14        printf("%d",sum);  
15        N = sum;  
16    }
```

And what is it that we did at this point? At this point, since I am going for the second round, the previous sum is incorrect any more. So, I need to initialize sum equals to 0 once more. And what I am going to do now is – I am going to put a while loop now. So, when will this happen? When do we need a while loop? If the number is greater than 9, that is when we will need a while loop. So, I did one iteration and the result was greater than 9. Only then, we will need a while loop. So, I am going to base this condition while n is greater than 9. Do everything here. So, I am going to adjust the indentation, so that you know what is happening. So, at this point, it is still not what the program expects; but let us try and compile it and run it.

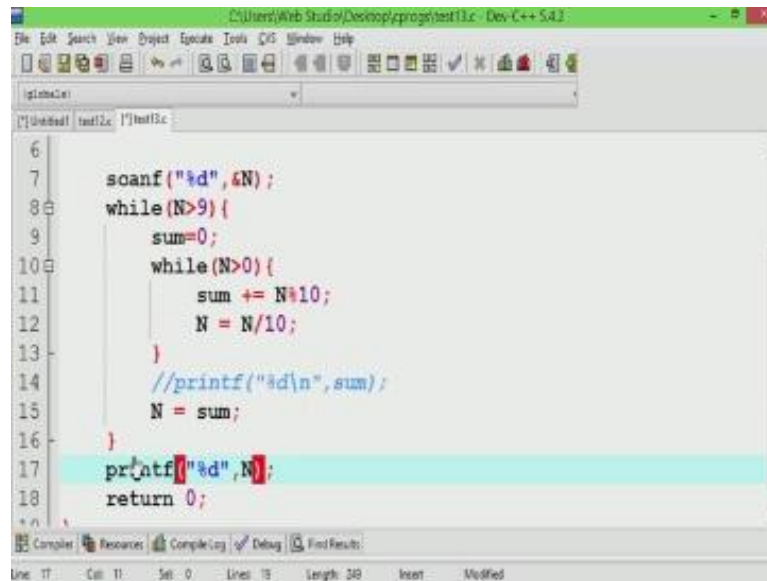
(Refer Slide Time: 09:57)



```
C:\Users\Web Studio\Desktop\prog\test13.c - [Executing] - Dev-C++ 5.4.2
C:\Users\Web Studio\Desktop\prog\test13.exe
969999999
69
15
6
-----
Process exited with return value 0
Press any key to continue . . .
```

So, I am going to try with 1234 to start with. So, 1234 gave a 101. So, that is actually 10 followed by a 1. So, to just see what the initial sums are, I am going to put a back slash n, so that we can see what the individual iterations give. So, I compile and run again. I give 1233; that is a 9 itself. Then, I give 1234; that gives a 10 followed by 1, which is good. 65536 – 25; and then, 7; which is again good. And 91123456; so five 9's; that is 55. So, 5 plus 5 is 10; 1 plus 0 is 1. So, it seems to be correct. And finally, 9612345 six 9's; that start with a 69 and then a 15 and then 6. It seems to be correct.

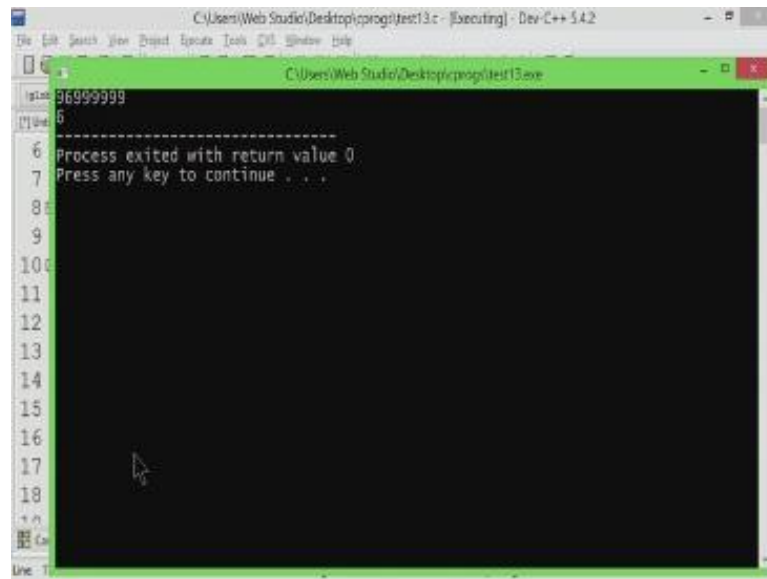
(Refer Slide Time: 10:55)



```
6  
7 scanf("%d", &N);  
8 while(N>9){  
9     sum=0;  
10    while(N>0){  
11        sum += N%10;  
12        N = N/10;  
13    }  
14    //printf("%d\n", sum);  
15    N = sum;  
16 }  
17 printf("%d", N);  
18 return 0;
```

So, what I am going to do now is it seems to be a program, which is correct except for this last thing that we are doing; this printf – we are printing everything. So, clearly, our program does not expect all of this. So, what I am going to do is – comment this line and put a printf here. So, I am going to print N here. So, if I put a printf here, then what it means is N is guaranteed to be less than or equal to 9; because otherwise, you would not have exited this while loop. So, let us try this.

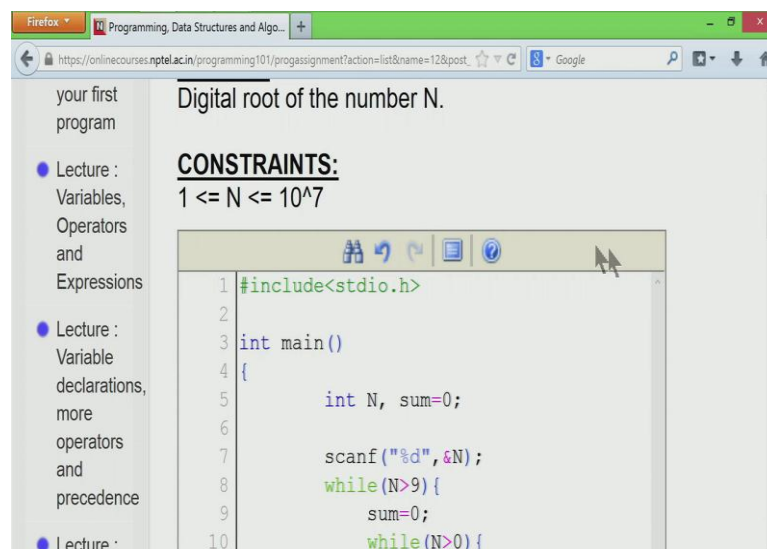
(Refer Slide Time: 11:34)



```
C:\Users\Web Studio\Desktop\prog\test13.c - [Executing] - Dev-C++ 5.4.2
C:\Users\Web Studio\Desktop\prog\test13.exe
96999999
6
-----
6 Process exited with return value 0
7 Press any key to continue . . .
8
9
10
11
12
13
14
15
16
17
18
```

So, again I will just give one test case and check 96123456; it gives me 6. So far, it seems right.

(Refer Slide Time: 11:47)



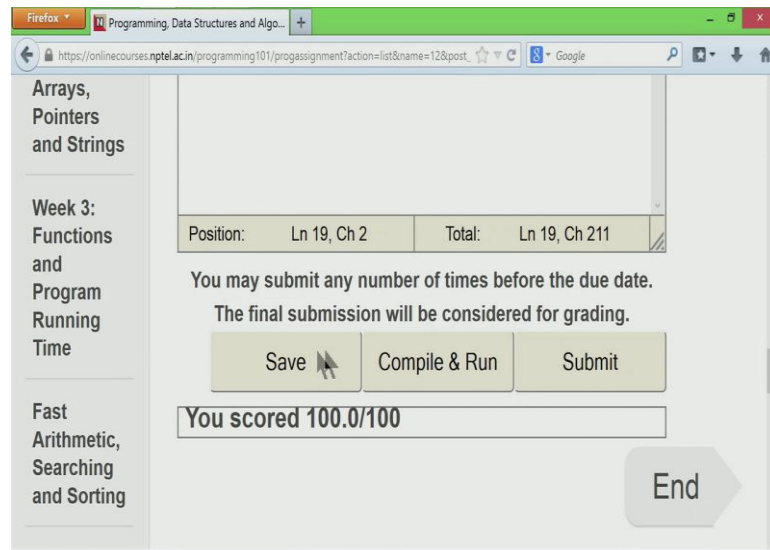
```
your first program
Lecture : Variables, Operators and Expressions
Lecture : Variable declarations, more operators and precedence
Lecture :

Digital root of the number N.
CONSTRAINTS:
1 <= N <= 10^7

1 #include<stdio.h>
2
3 int main()
4 {
5     int N, sum=0;
6
7     scanf("%d", &N);
8     while(N>9) {
9         sum=0;
10        while(N>0) {
```

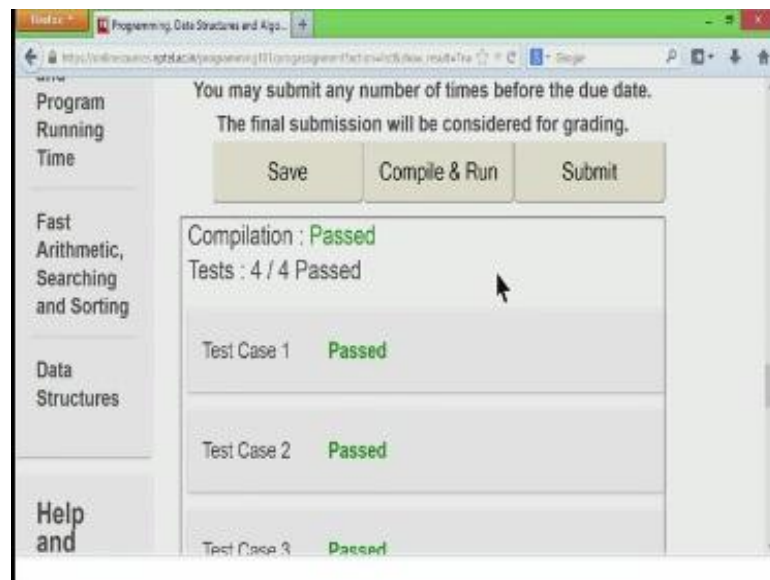
So, I am going to take this program. So, I am cutting and pasting it into the online portal. So, this is the online portal; I start with a black template and I am going to put it here.

(Refer Slide Time: 11:57)



I am cutting and pasting it. And I will save, and compile and run.

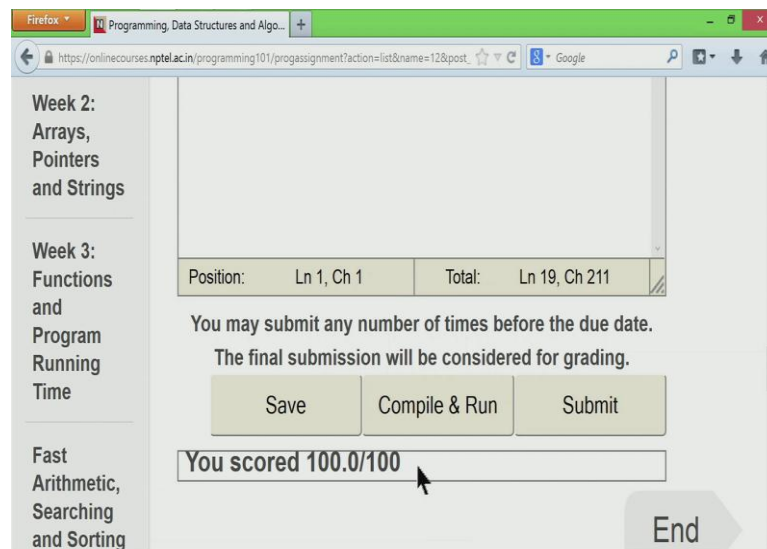
(Refer Slide Time: 12:12)



So, when I compile and run, let me see the results. I seem to the passed all the test cases. And if I have passed all the private test cases... Or, the public test cases as you see here; whatever you see – given to you. So, we have two kinds of test cases: one is called

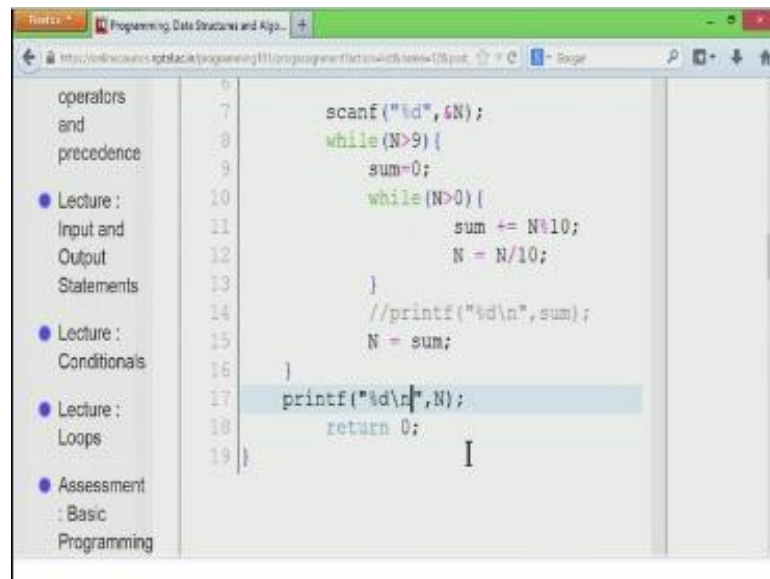
public test case; and another is called private test case. The public test cases are shown to you; the private test cases are not shown to you. When I submit; it actually evaluates against a private test cases, which are not shown to you.

(Refer Slide Time: 12:39)



And let me see whether I have got 100. So, I seem to have 100 on 100. So, in which case, this program seems to be correct.

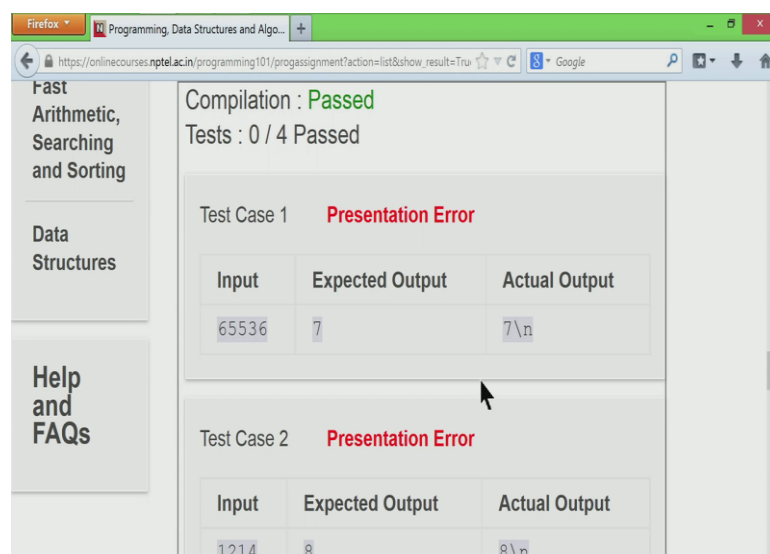
(Refer Slide Time: 12:50)



```
scanf("%d", &N);
while(N>9){
    sum=0;
    while(N>0){
        sum += N%10;
        N = N/10;
    }
    //printf("%d\n", sum);
    N = sum;
}
printf("%d\n", N);
return 0;
```

So, one subtle thing that I want to show here is this other notion of... What if I put an extra back slash n or things like that here. So, let us say I put a back slash n, which is extra new line; this is something that we seem to do many times. So, let us say I put a back slash n and I compile and run.

(Refer Slide Time: 13:08)

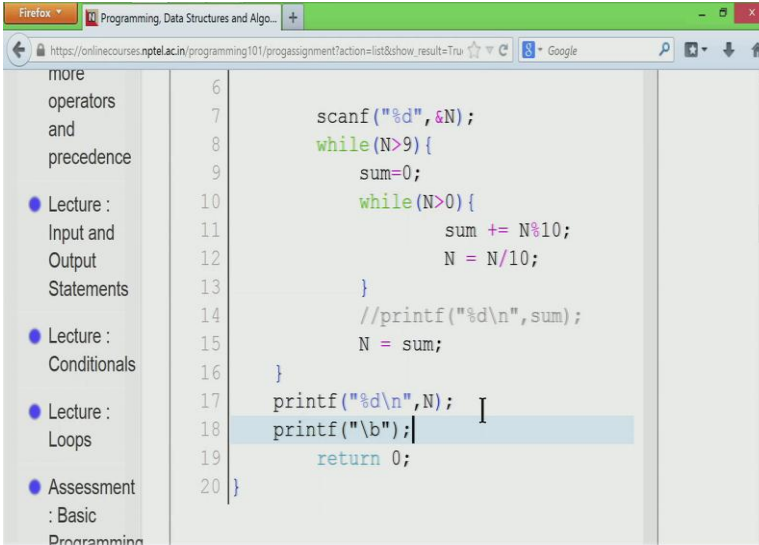


Compilation : **Passed**
Tests : 0 / 4 Passed

Test Case	Input	Expected Output	Actual Output
Test Case 1	65536	7	7\n
Test Case 2	1214	8	8\n

We get... We would get what is called the presentation error. So, this is something that many of you posted on the forum that, your output is correct; but there is something called presentation error. So, actually it clearly says what a presentation error is. So, if the expected output is 7; and instead we have printed 7 back slash n.

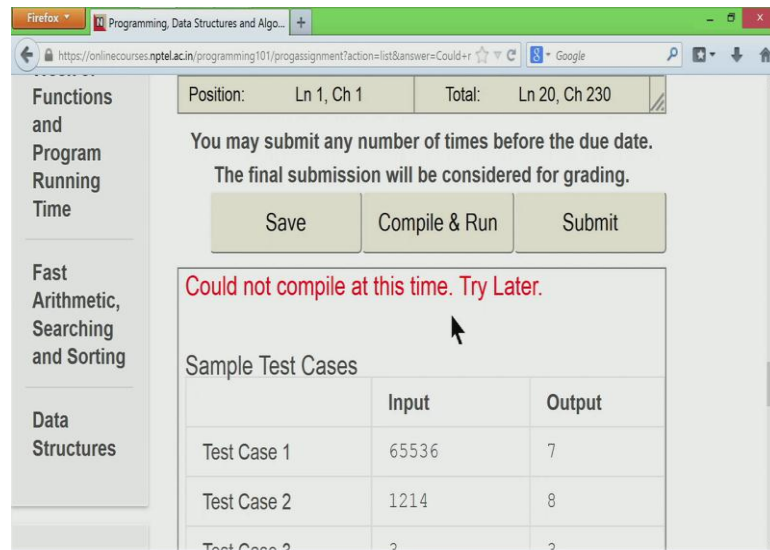
(Refer Slide Time: 13:29)



```
6
7     scanf("%d", &N);
8     while(N>9) {
9         sum=0;
10        while(N>0) {
11            sum += N%10;
12            N = N/10;
13        }
14        //printf("%d\n", sum);
15        N = sum;
16    }
17    printf("%d\n", N);
18    printf("\b");
19    return 0;
20 }
```

So, what it really says is just get rid of the back slash n. And it seems like many of you are doing this. You want to get rid of the back slash n. So, instead of removing it from the printf, you seem to be doing this – put a back slash b, which stands for back space.

(Refer Slide Time: 13:45)

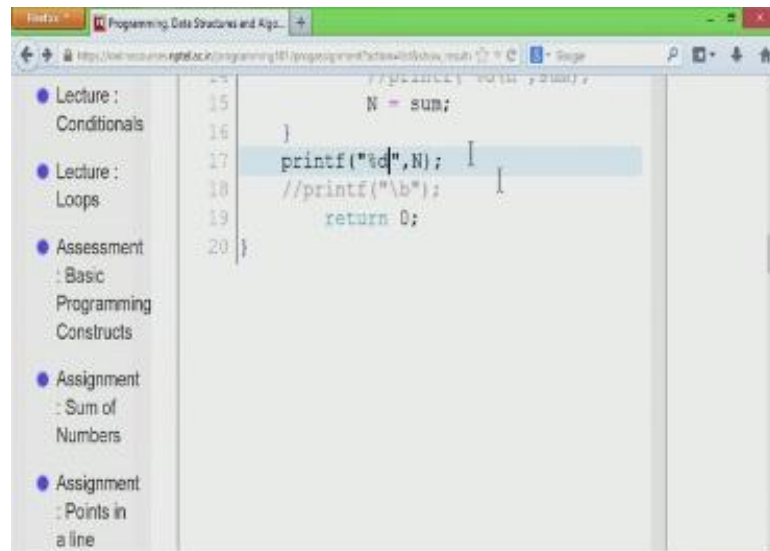


The screenshot shows a web browser window with the URL <https://onlinecourses.nptel.ac.in/programming101/progassignment?action=list&answer=Could+r>. The page displays submission options: "Save", "Compile & Run", and "Submit". Below these buttons, a red error message reads: "Could not compile at this time. Try Later." A mouse cursor is positioned over the error message. Below the error message is a table titled "Sample Test Cases" with columns for "Input" and "Output".

	Input	Output
Test Case 1	65536	7
Test Case 2	1214	8
Test Case 3	3	3

So, let us say I did that; I do a compile and run on that; I get this really bizarre error. It says could not compile at this time. Try later. So, this back slash b is not something that is supported by the compiler to be displayed properly. And the compilation actually fails here. So, it may sound bizarre to you, instead of giving it to you as an error; the compilation itself seems to be failing. But technically, it is not the compiler which fails; it is just that it is a bizarre way of giving this error out.

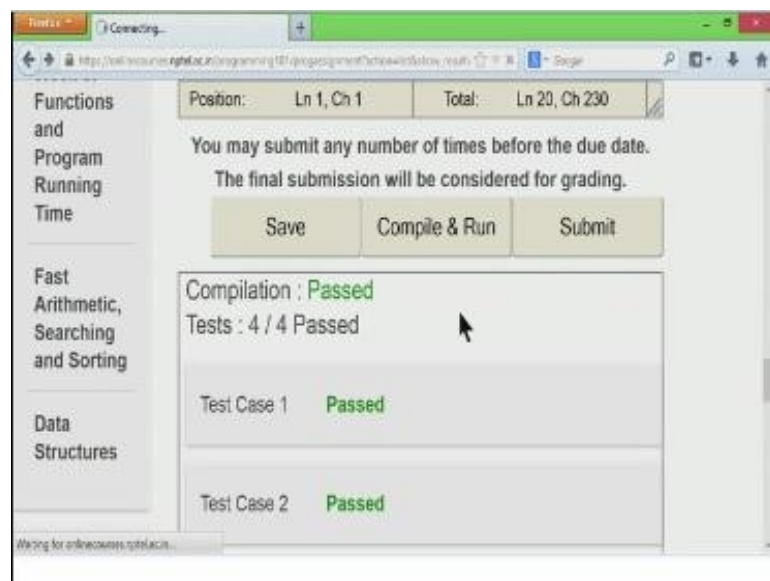
(Refer Slide Time: 14:18)



```
//printf("%d\n", N);
N = sum;
}
printf("%d", N);
//printf("%b");
return 0;
}
```

So, instead, I am not going to murkier on with this; what I am going to do is I am just going to comment this line and get rid of the back slash n. So, once I get rid of the back slash n, things a fine.

(Refer Slide Time: 14:25)



Position: Ln 1, Ch 1 Total: Ln 20, Ch 230

You may submit any number of times before the due date.
The final submission will be considered for grading.

Save Compile & Run Submit

Compilation : **Passed**
Tests : 4 / 4 Passed

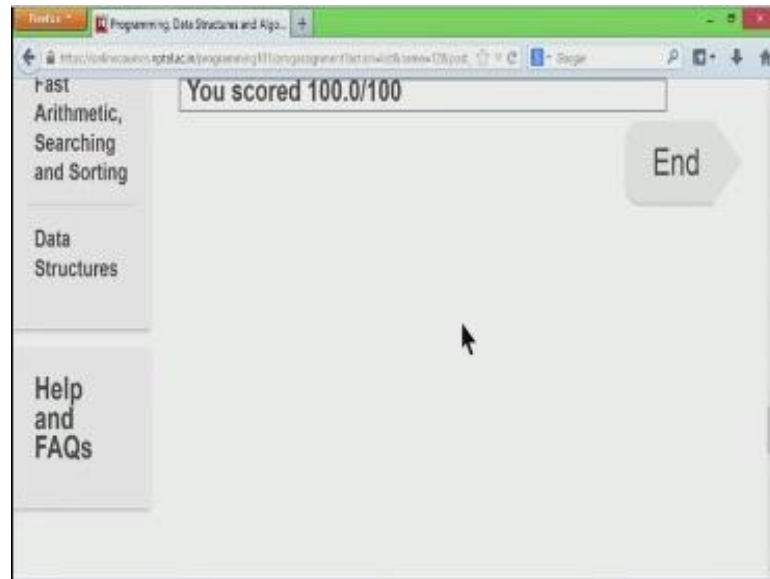
Test Case 1 **Passed**

Test Case 2 **Passed**

Waiting for onlinecourses.opit.ac.in

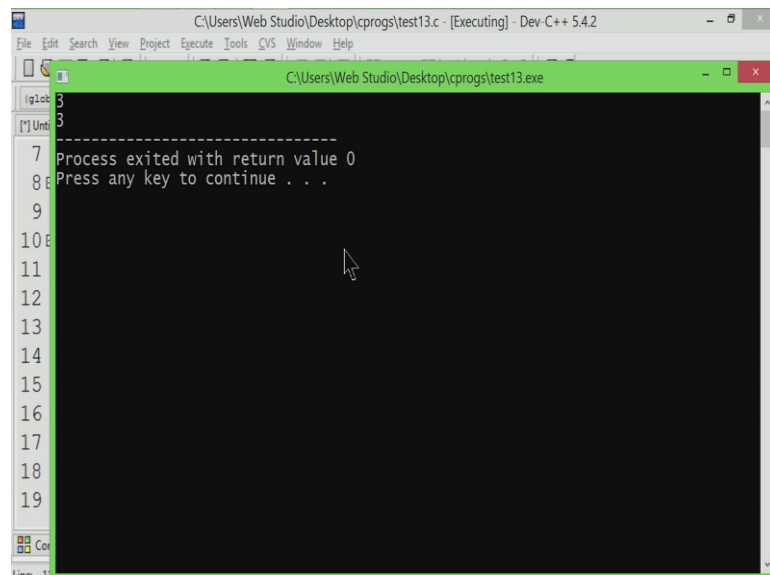
So, I compile and run it. I passed all the test cases; I submit. And this is OK.

(Refer Slide Time: 14:40)



So, I would get 100 on 100. So, one nice thing that this program has is if the number is... So, this is the test case that I did not talk to you about.

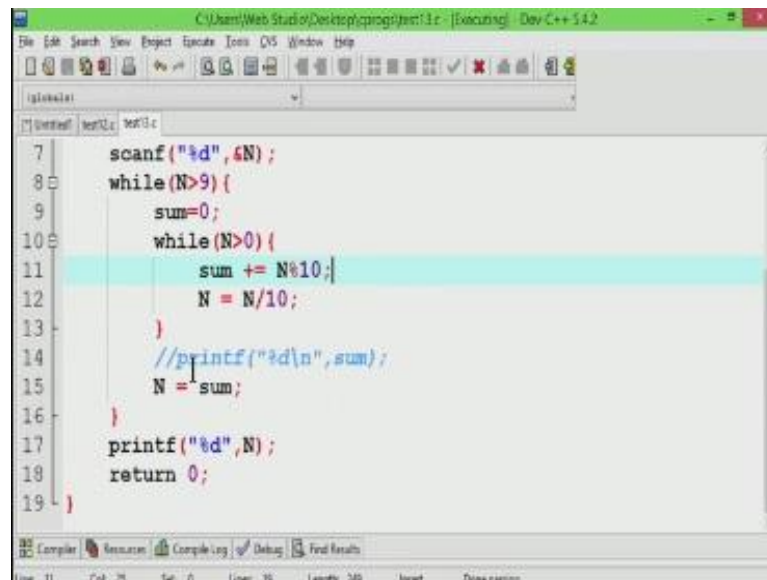
(Refer Slide Time: 14:48)



So, let us say I give you this number. I test it with 3; it is a single digit number; which means you should not do any more processing. So, it should exit with 3 itself. And that is

something that, the program automatically takes care of.

(Refer Slide Time: 15:01)



```
7 scanf("%d", &N);
8 while(N>9){
9     sum=0;
10    while(N>0){
11        sum += N%10;
12        N = N/10;
13    }
14    //printf("%d\n", sum);
15    N = sum;
16 }
17 printf("%d", N);
18 return 0;
19 }
```

So, look at this. If the very first time you get N; itself is less than or equal to 9; then, you do not execute this while loop; you directly print the number. So, clearly, this program will work correctly only if you have numbers, which are positive. For negative numbers, it will print the negative number as it is; and digital root is not defined for negative number. So, for any number that is positive, it is finding out the correct digital sum. So, this is just a small video that I wanted to show you on how to solve this problem and how to write code. And I am hoping that, this will help you in breaking your barrier and being able to translate ideas into code. We are hoping to also actually show such sessions for the subsequent weeks. So, we will take some of these slightly difficult problems and show you how to solve the problem and also how to write programs with it.

So, thank you very much. And I wish you all the best for this course. I am sure that, you are enjoying this course. There are a few main points that you see with respect to presentation and so on. So, overall, I hope that, you are enjoying the course.

Thank you and bye.