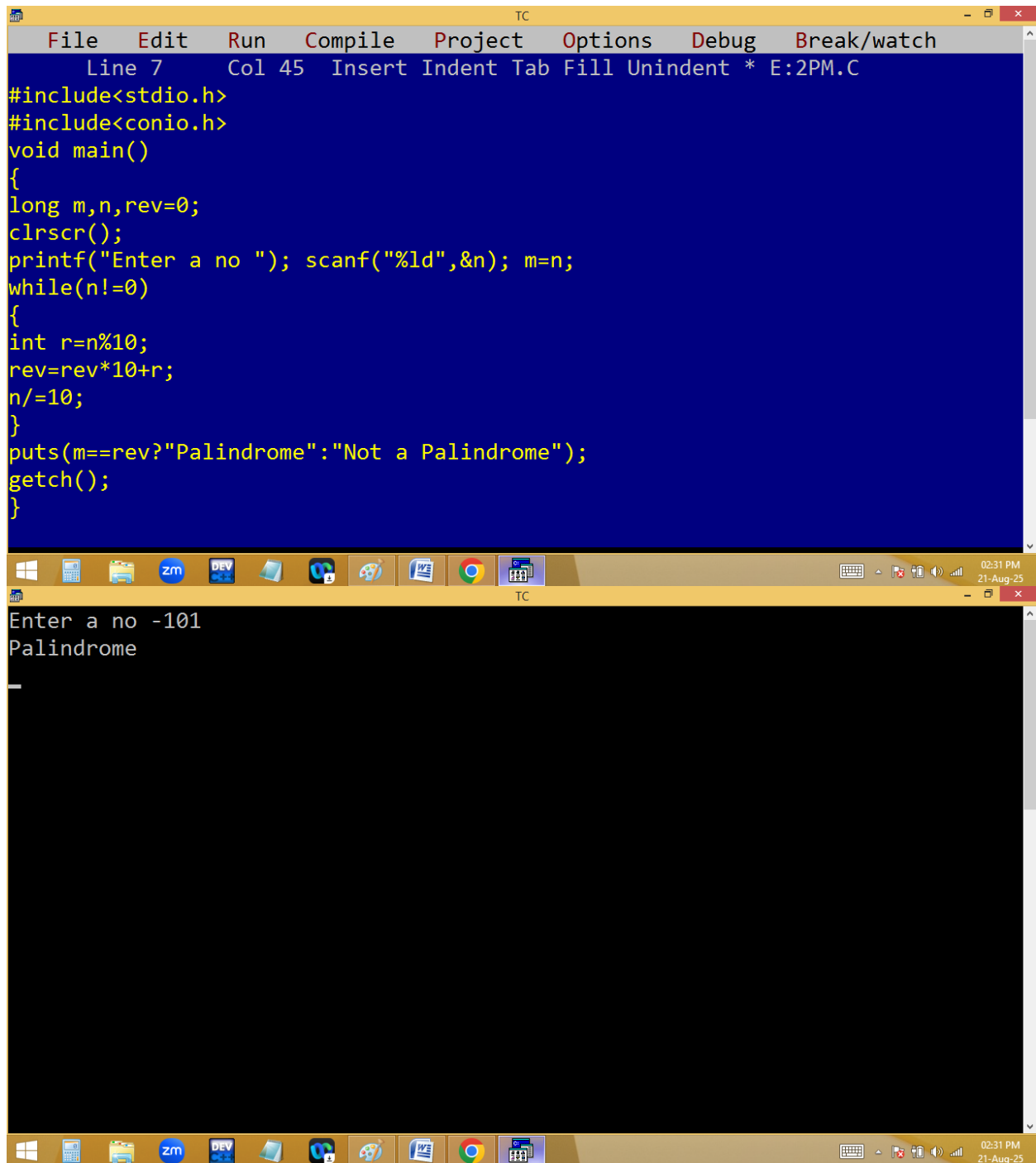


Finding palindrome no?

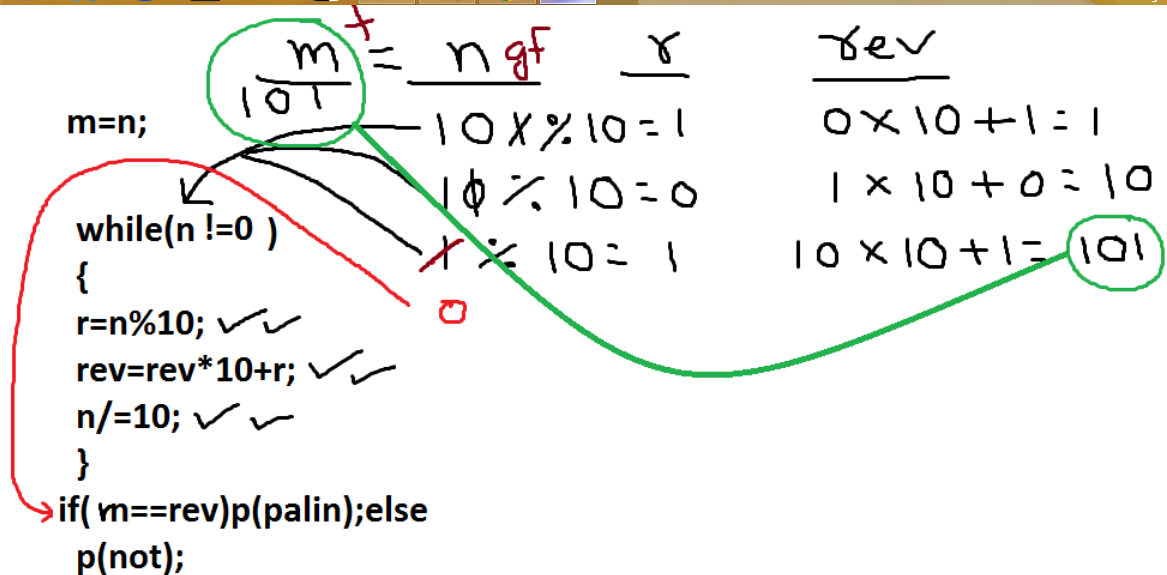
Given no reverse no both are same



```
File Edit Run Compile Project Options Debug Break/watch
Line 7 Col 45 Insert Indent Tab Fill Unindent * E:2PM.C
#include<stdio.h>
#include<conio.h>
void main()
{
long m,n,rev=0;
clrscr();
printf("Enter a no "); scanf("%ld",&n); m=n;
while(n!=0)
{
int r=n%10;
rev=rev*10+r;
n/=10;
}
puts(m==rev?"Palindrome":"Not a Palindrome");
getch();
}
```

Enter a no -101
Palindrome

```
TC
Enter a no 123
Not a Palindrome
```



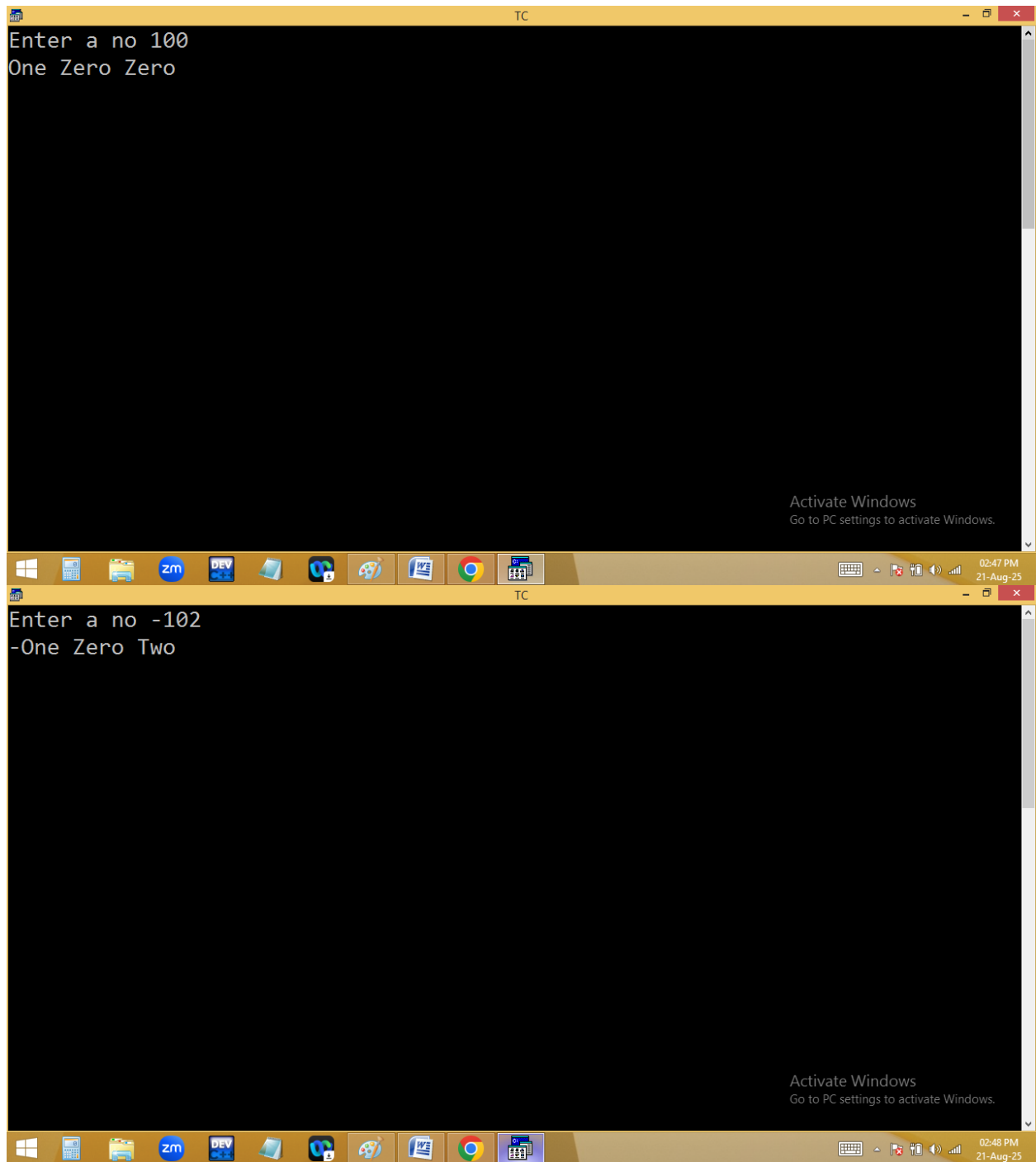
Number to text conversion?

```
#include<stdio.h>
```

```
#include<conio.h>
```

```
void main()  
  
{  
  
long m,n,rev=0;  
  
clrscr();  
  
printf("Enter a no "); scanf("%ld",&n);  
  
if(n<0)printf("-",n=-n); m=n;  
  
while(m!=0){int  
r=m%10;rev=rev*10+r;m/=10;}/*rev*/  
  
do  
  
{  
  
switch(rev%10)  
  
{  
  
case 0: printf("Zero");break;  
  
case 1: printf("One");break;  
  
case 2: printf("Two");break;  
  
case 3: printf("Three");break;
```

```
case 4: printf("Four");break;
case 5: printf("Five");break;
case 6: printf("Six");break;
case 7: printf("Seven");break;
case 8: printf("Eight");break;
case 9: printf("Nine");break;
}
rev=rev/10; printf(" ");
}while(rev!=0);
while(n%10==0 && n!=0) printf("Zero ",n/=10);
getch();
}
```



102 → One Zero Two

$\frac{n}{102} \quad \frac{m}{102} \rightarrow \frac{rev}{207 \div 10 = 1}$

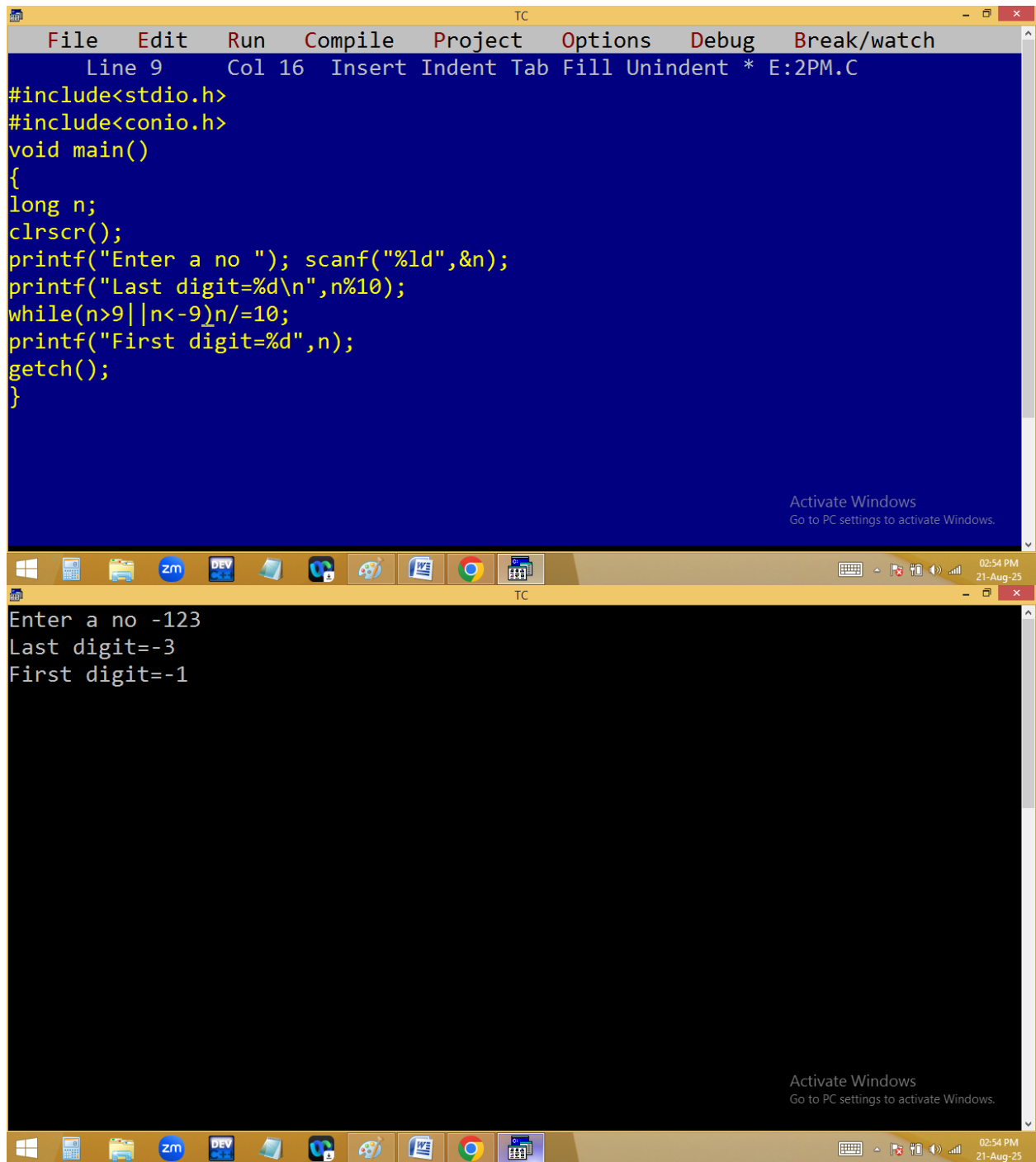
```

do
{
switch(rev%10)
{
case 0: p("Zero");break;
case 1: p("One");break;
case 9: p("Nine");break;
}
rev=rev/10;
}while(rev!=0);

```

$207 \div 10 = 0$ Zero ✓
 $207 \div 10 = 2$ two ✓
 0 One zero zero
 $\frac{n}{107 \div 10 = 0}$
 $\text{while}(n\%10==0 \& \& n!=0) p(\text{"zero"}, n/=10);$

Printing 1st and last digits of given no



The image shows two screenshots of the Turbo C++ (TC) IDE. The top screenshot displays the source code of a C program in a blue editor window. The code is as follows:

```
File Edit Run Compile Project Options Debug Break/watch
Line 9 Col 16 Insert Indent Tab Fill Unindent * E:2PM.C
#include<stdio.h>
#include<conio.h>
void main()
{
long n;
clrscr();
printf("Enter a no "); scanf("%ld",&n);
printf("Last digit=%d\n",n%10);
while(n>9||n<-9)n/=10;
printf("First digit=%d",n);
getch();
}
```

The bottom screenshot shows the same IDE with the program's output in a black window. The output is:

```
Enter a no -123
Last digit=-3
First digit=-1
```

Both screenshots include a Windows taskbar at the bottom with various application icons and a system tray showing the time as 02:54 PM on 21-Aug-25. An "Activate Windows" watermark is visible in the bottom right corner of both windows.

```
TC
Enter a no 9
Last digit=9
First digit=9_

Activate Windows
Go to PC settings to activate Windows.
```

```
TC
Enter a no 75623
Last digit=3
First digit=7_

Activate Windows
Go to PC settings to activate Windows.
```

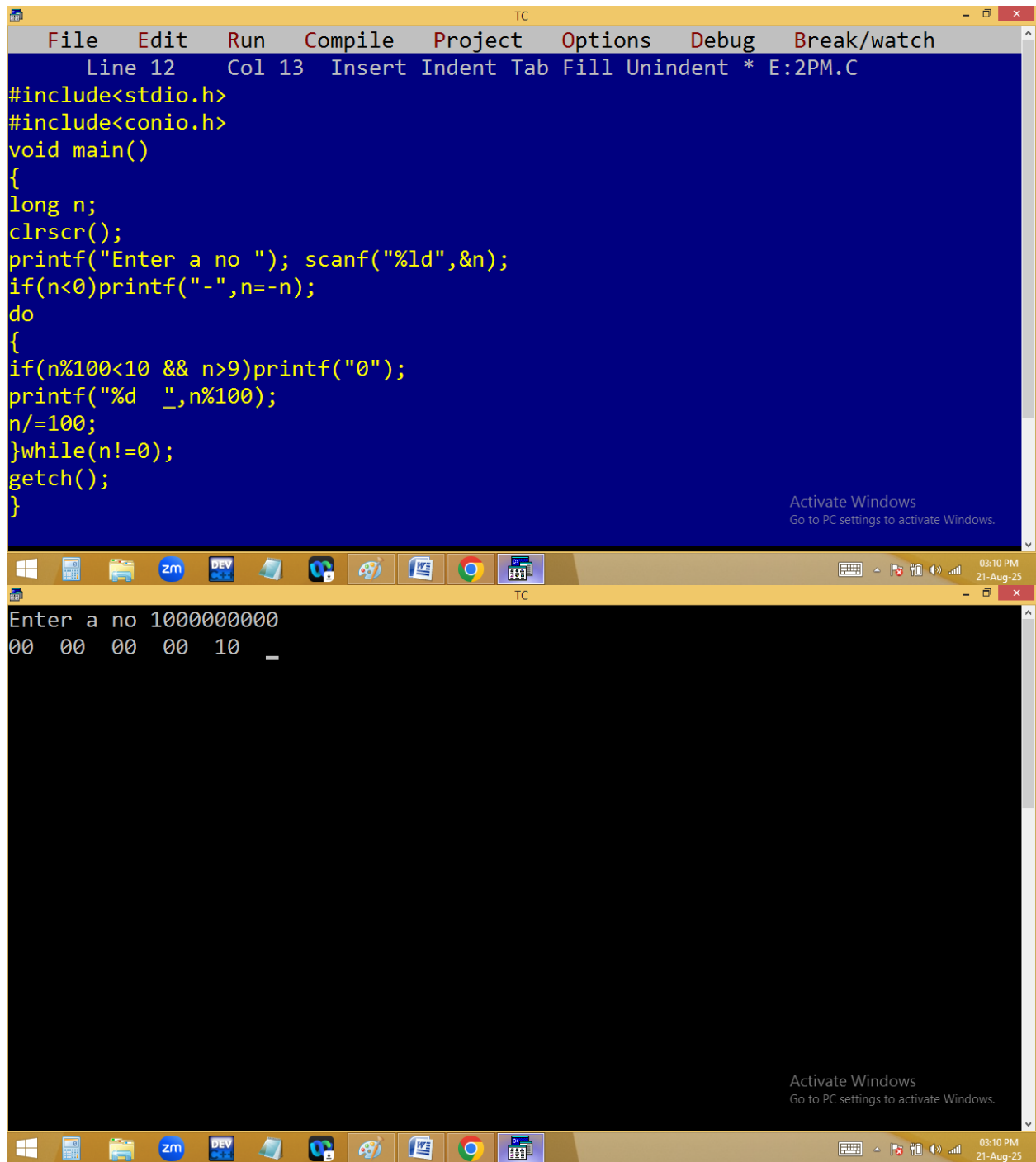

p("Last digit=%d",n%10);

$$\begin{array}{r} n \\ \hline 123 \div 10 = 3 \end{array}$$

while(n>9) n/=10; ✓

p("First digit=%d",n);

12345678 → 78 56 34 12



The image shows a screenshot of the Turbo C++ (TC) IDE. The top window displays the source code for a C program named E:2PM.C. The code is as follows:

```
Line 12 Col 13 Insert Indent Tab Fill Unindent * E:2PM.C
#include<stdio.h>
#include<conio.h>
void main()
{
long n;
clrscr();
printf("Enter a no "); scanf("%ld",&n);
if(n<0)printf("-",n=-n);
do
{
if(n%100<10 && n>9)printf("0");
printf("%d ",n%100);
n/=100;
}while(n!=0);
getch();
}
```

The bottom window shows the execution output. The user has entered the number 1000000000. The program has processed this number and displayed the output: 00 00 00 00 10, followed by a cursor. The output represents the digits of the input number in reverse order, with leading zeros added for the first four digits.

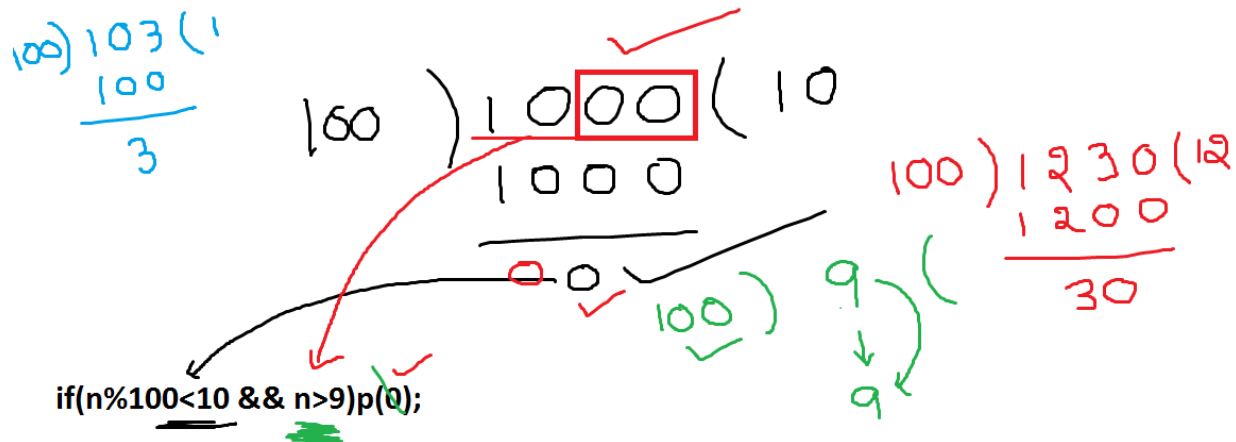
03:10 PM
21-Aug-25

```
TC
Enter a no -123400
-00 34 12 _
```

Activate Windows
Go to PC settings to activate Windows.

```
TC
Enter a no 9
9 _
```

Activate Windows
Go to PC settings to activate Windows.



Finding Armstrong no?

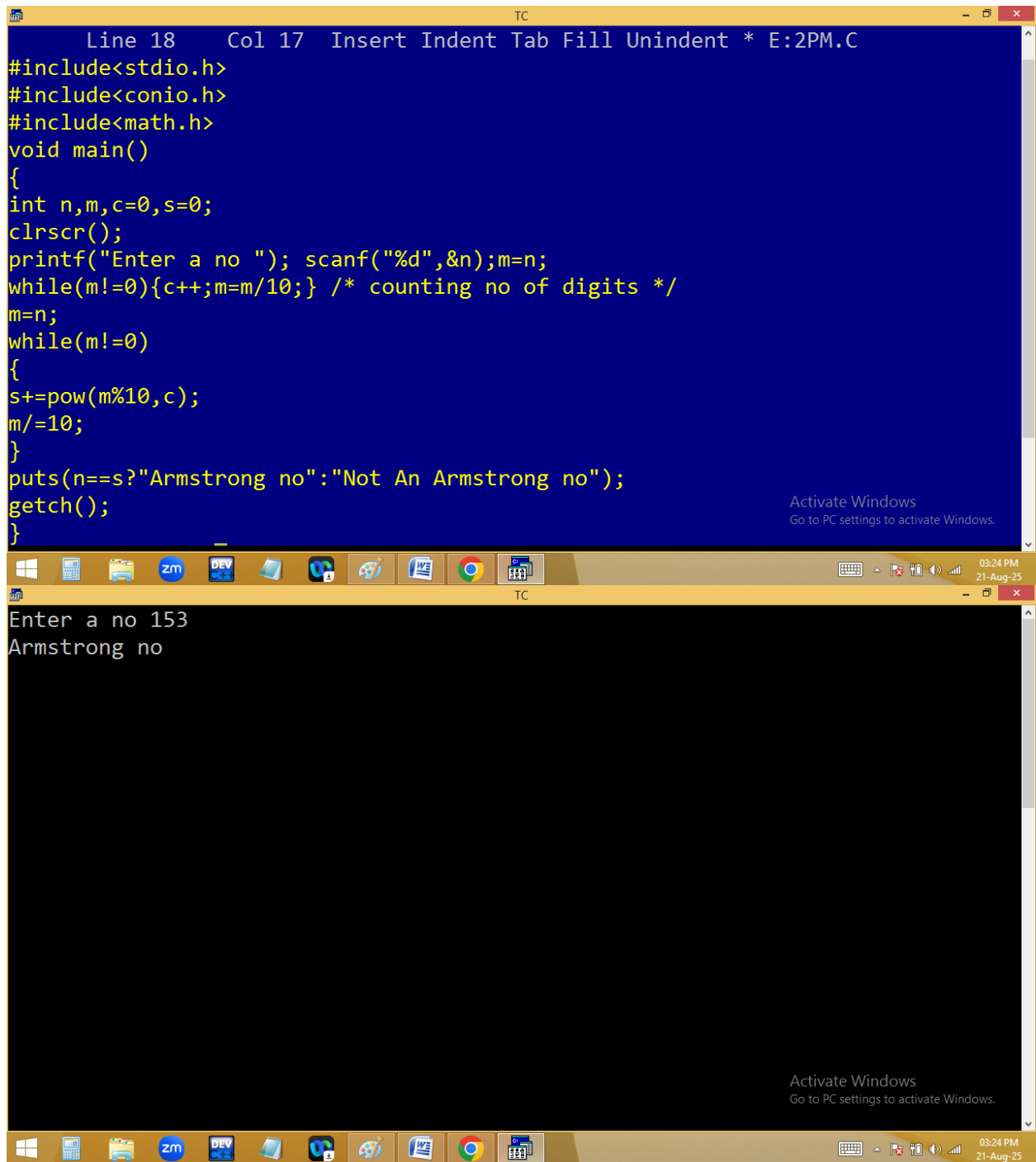
1 is a single digit no $\rightarrow 1^1 = 1$

2 is a single digit no $\rightarrow 2^1 = 2$

9 is a single digit no $\rightarrow 9^1 = 9$

153 is a three digit no $\rightarrow 1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$

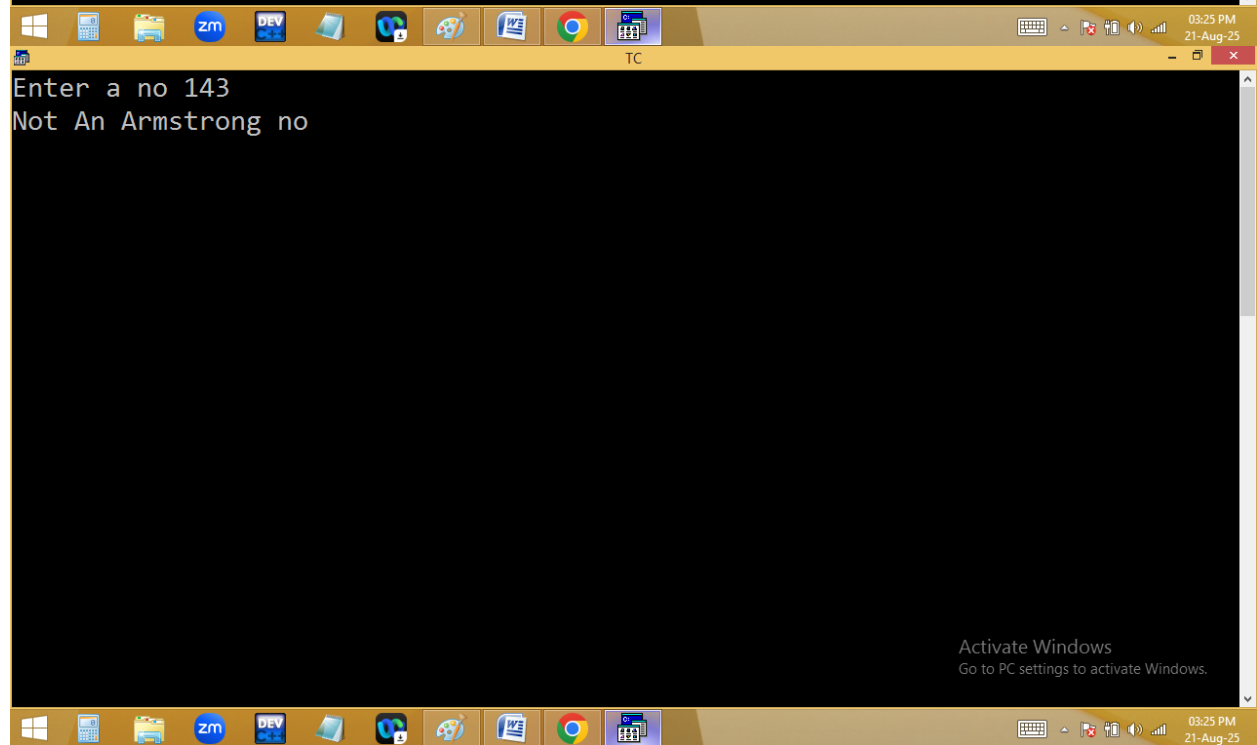
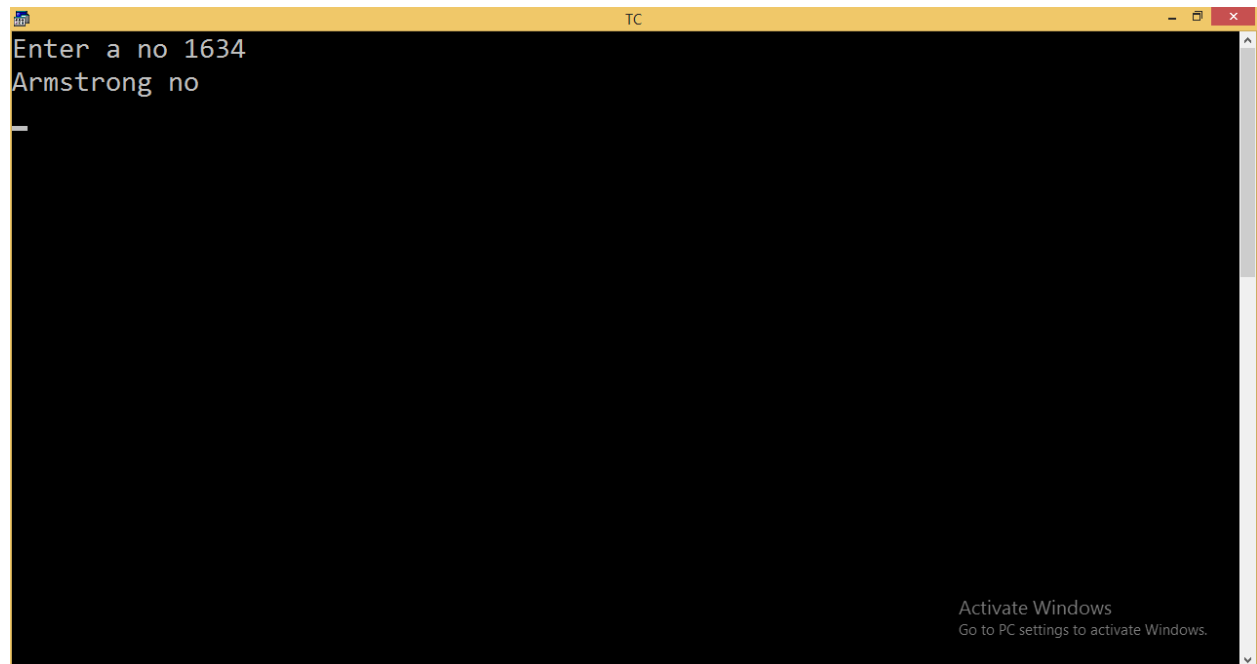
Eg: 370, 371, 407, 1634, 8208,....

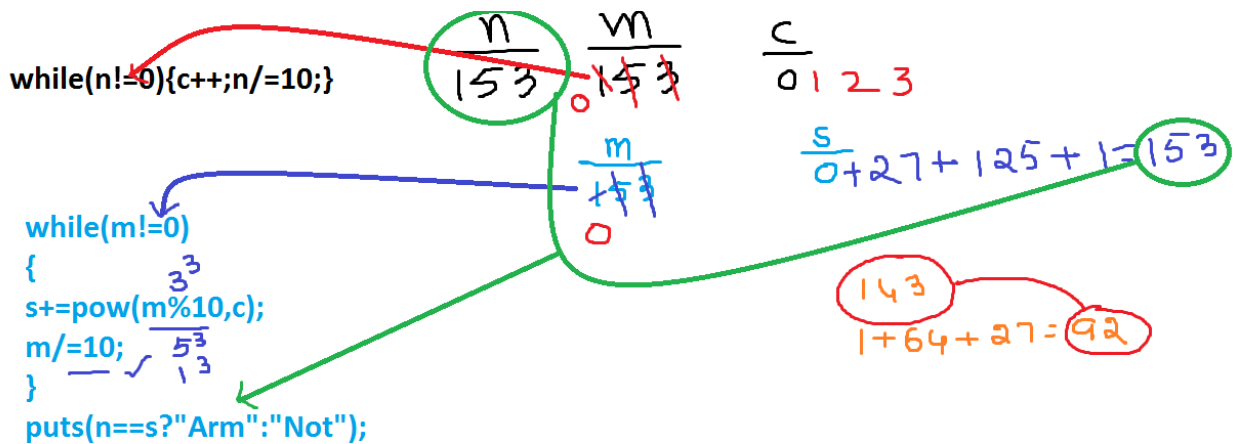


The image shows a Windows desktop with two instances of the Turbo C++ (TC) compiler. The top window, titled 'TC', displays the source code for a program that checks if a number is an Armstrong number. The code is as follows:

```
Line 18   Col 17   Insert Indent Tab Fill Unindent * E:2PM.C
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
int n,m,c=0,s=0;
clrscr();
printf("Enter a no "); scanf("%d",&n);m=n;
while(m!=0){c++;m=m/10;} /* counting no of digits */
m=n;
while(m!=0)
{
s+=pow(m%10,c);
m/=10;
}
puts(n==s?"Armstrong no":"Not An Armstrong no");
getch();
}
```

The bottom window, also titled 'TC', shows the program's execution. It prompts the user to 'Enter a no' and the input '153' is provided. The output of the program is 'Armstrong no'. Both windows include a taskbar at the bottom with various application icons and a system tray showing the time as 03:24 PM on 21-Aug-25. An 'Activate Windows' watermark is visible in the bottom right corner of both windows.





for loop:

It is an entry control loop.

for is a keyword.

It is also used to repeat a program several times based on a condition.

When compared with while and do while, for loop is looking to be smart. In for it is compulsory to maintain two semicolons.

For works without condition also and default condition is always 1 i.e. true.

Generally for loop is having 3 expressions.

- 1. Initialization**
- 2. Test condition / expression**
- 3. Increment/decrement / updation**

At first entry of for loop the initialization part is executed and later the test condition is checked. If the condition is true then the for block statements are executed. After completion of the block, the increment or decrement part is executed. Later once again the test condition is evaluated. If it is true then once again for block statements are executed. Like this the process is continued until the condition becomes false. Here the

initialization part is executed only once, at the time of loop beginning.

It is mandatory to maintain 2 semicolon (;) in a for loop.

If the for loop is having more than three expressions, it is mandatory to separate the expressions with , separator.

If the for loop is having less than three expressions, then leave the expressions with empty semicolon.

