

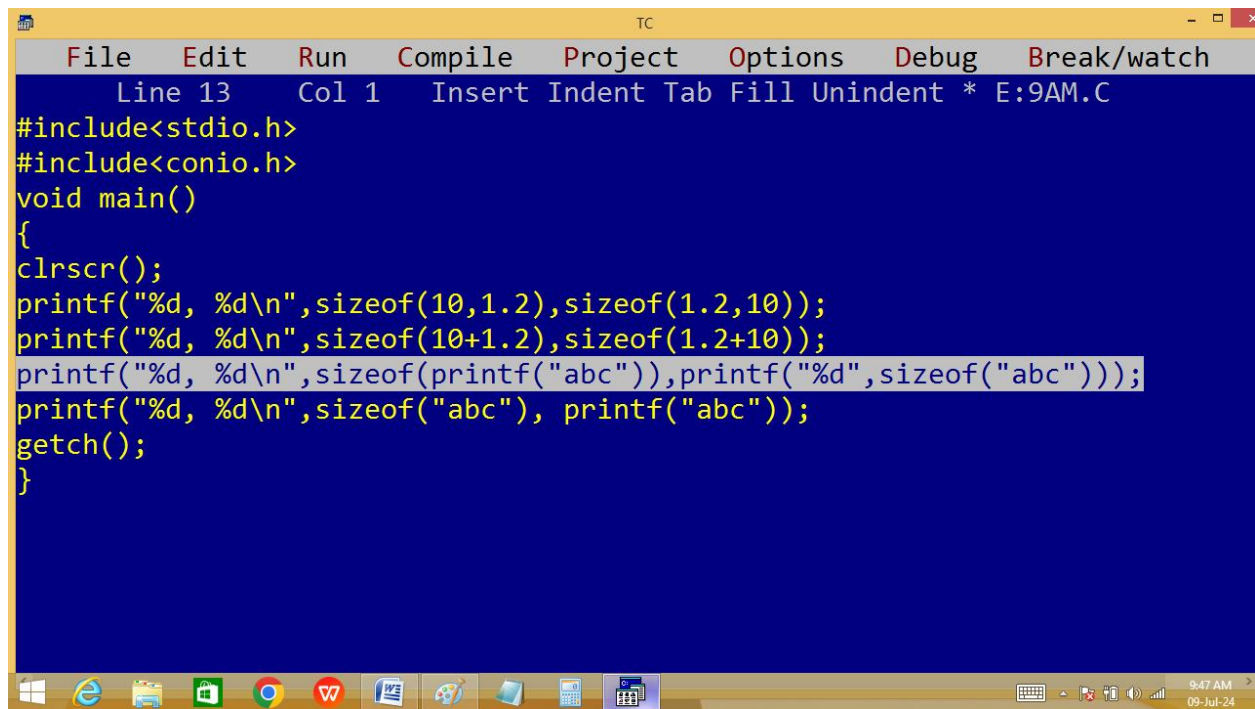
The image shows a screenshot of the Turbo C++ (TC) IDE. The top window displays a C program with the following code:

```
File Edit Run Compile Project Options Debug Break/watch
Line 11 Col 55 Insert Indent Tab Fill Unindent * E:9AM.C
#include<stdio.h>
#include<conio.h>
void main()
{
char s1[10]="Hyd", s2[]="Hyd";
clrscr();
printf("%d, %d\n",sizeof(s1),sizeof(s2));
printf("%d, %d\n",sizeof("Hyd"),sizeof("Kishore Naidu"));
printf("abc address = %u\n","abc");
printf("%d, %d\n",sizeof("1.23"),sizeof("abc"+1));
printf("%d, %d\n",sizeof("abc")+1,sizeof(sizeof("abc")));
getch();
}
```

The bottom window shows the output of the program:

```
10, 4
4, 14
abc address = 452
5, 2
5, 2
```

The IDE interface includes a menu bar with options like File, Edit, Run, Compile, Project, Options, Debug, and Break/watch. The status bar at the bottom right shows the time as 9:28 AM and the date as 09-Jul-24.



```
File Edit Run Compile Project Options Debug Break/watch
Line 13 Col 1 Insert Indent Tab Fill Unindent * E:9AM.C
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
printf("%d, %d\n",sizeof(10,1.2),sizeof(1.2,10));
printf("%d, %d\n",sizeof(10+1.2),sizeof(1.2+10));
printf("%d, %d\n",sizeof(printf("abc")),printf("%d",sizeof("abc")));
printf("%d, %d\n",sizeof("abc"), printf("abc"));
getch();
}
```

```
TC
8, 2
8, 8
42, 1
abc4, 3
```

sizeof(sizeof("abc"))

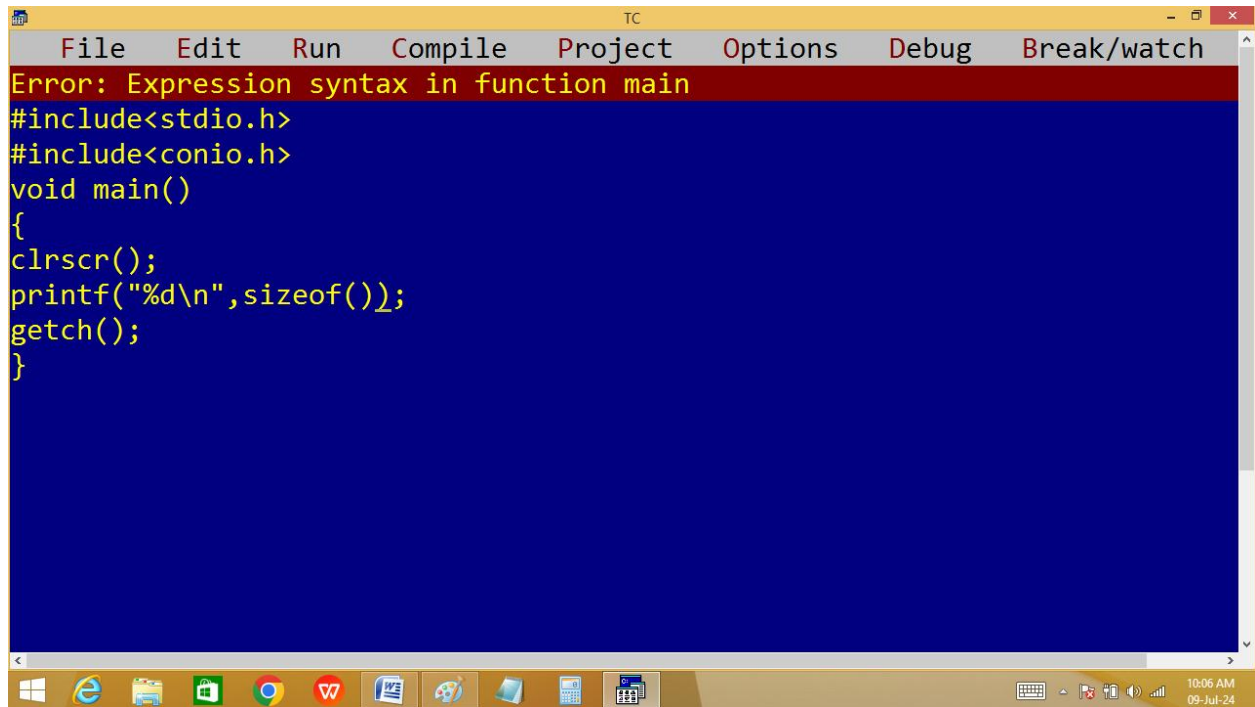
sizeof (4)
2

printf("%d, %d", sizeof("abc"), printf("abc"));



```
TC
Line 17 Col 18 Insert Indent Tab Fill Unindent * E:9AM.C
#include<stdio.h>
#include<conio.h>
void main()
{
int a=9999;
clrscr();
printf("%d, %d\n",sizeof(" "),sizeof(""));
printf("%d, %d\n",sizeof("abc\0"),printf("abc\0"));
printf("%d\n",sizeof(++a));
printf("a=%d\n",a);
printf("%d\n",sizeof(a=1.5));
printf("a=%d\n",a);
printf("%d\n",sizeof(a=5555));
printf("a=%d",a);
getch();
}
/* Inside sizeof( ) expressions not considered */
```

```
TC
2, 1
abc5, 3
2
a=9999
2
a=9999
2
a=9999_
```

A screenshot of the Turbo C++ (TC) IDE. The window title is 'TC'. The menu bar includes 'File', 'Edit', 'Run', 'Compile', 'Project', 'Options', 'Debug', and 'Break/watch'. A red error message banner at the top reads 'Error: Expression syntax in function main'. The code editor has a blue background and contains the following C code:

```
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
printf("%d\n",sizeof());
getch();
}
```

The Windows taskbar is visible at the bottom with various application icons and a system clock showing 10:06 AM on 09-Jul-24.

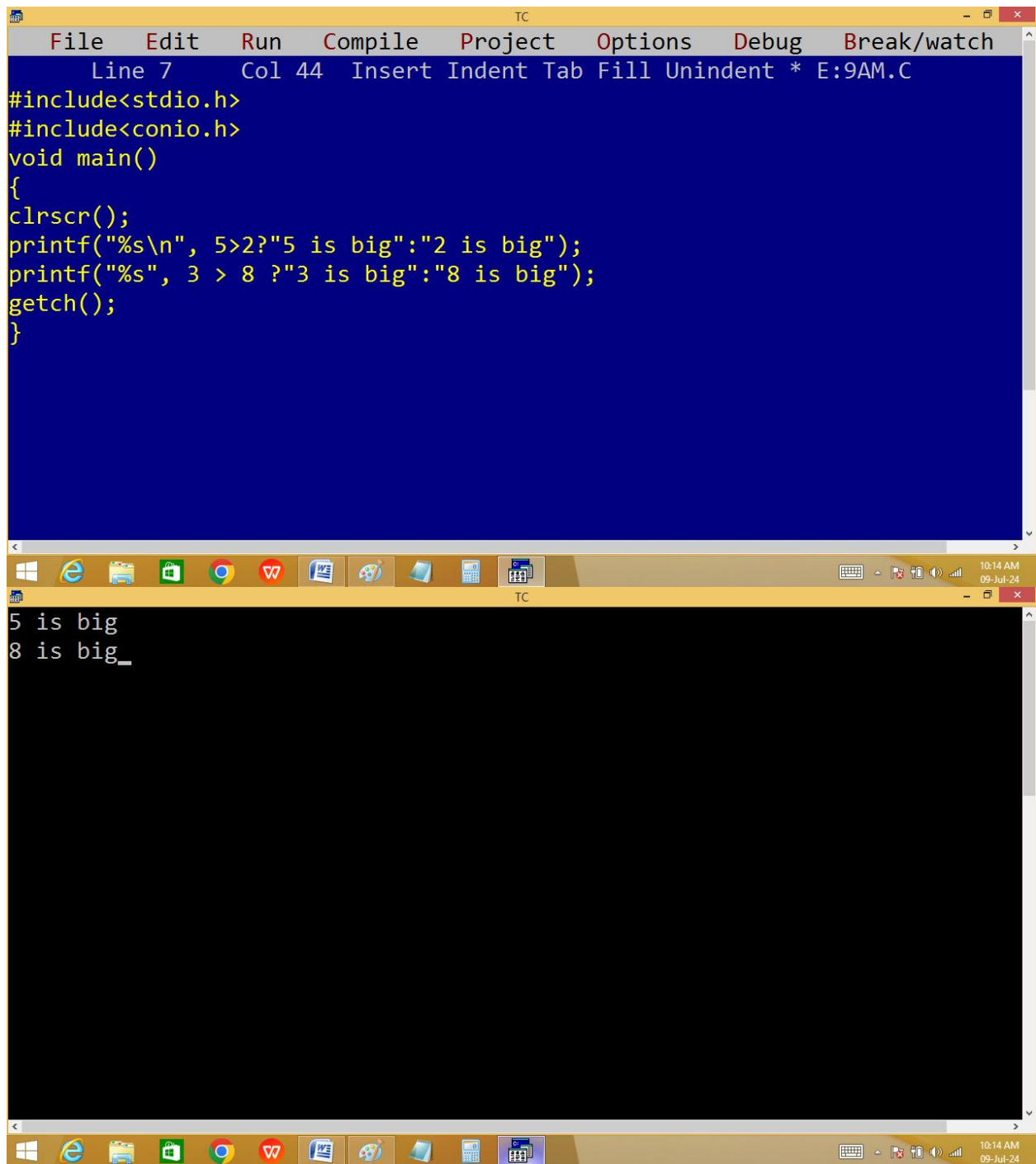
Ternary operator / conditional operator [?:]

Conditional part ? true part : false part ;

Eg:

5 > 2 ? "5 is big" : "2 is big";

3 > 8 ? "3 is big" : "8 is big";



The image shows a screenshot of the Turbo C++ (TC) IDE. The top window displays a C program with the following code:

```
File Edit Run Compile Project Options Debug Break/watch
Line 7 Col 44 Insert Indent Tab Fill Unindent * E:9AM.C
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
printf("%s\n", 5>2?"5 is big":"2 is big");
printf("%s", 3 > 8 ?"3 is big":"8 is big");
getch();
}
```

The bottom window shows the output of the program:

```
5 is big
8 is big_
```

The Windows taskbar at the bottom shows the time as 10:14 AM on 09-Jul-24.

BITWISE OPERATORS

Bitwise operator's works on bits.

Turbo-c is a 16 bit compiler. Due to this bitwise operations are limited to 16 bits only [2^0 to 2^{15}].

Bitwise operators operate **integer** type values only.

We have to calculate only the **on** bits [**1**].

When the first bit[**Sign bit**] is **1** then the number is **Negative** and it is **0** then the number is **positive**.

They are very much used in system software development.

Note: Bitwise operator is low level feature.

C-Language supports following bitwise operators.

& -Bitwise and

| - Bitwise or

^ - XOR ==> Exclusive OR

~ - Compliment operator

<< - Left shift operator

>> - Right shift operator

& - Bitwise and: In this both bits are 1's then result bit is 1. Otherwise result bit is 0.

Eg: **25 & 15 = 9**

25 = 0000 0000 0001 1001
15 = 0000 0000 0000 1111

2 | 25
2 | 12 - 1
2 | 6 - 0
2 | 3 - 0
1 - 1

2 | 15
2 | 7 - 1
2 | 3 - 1
1 - 1

25 & 15 = 9

25 = 0000 0000 0001 1001
15 = 0000 0000 0000 1111

&

0000 0000 0000 1001

↓ ↓
 $2^3 + 2^0$
↓ ↓
8 + 1 = 9

| - Bitwise or: In this both bits are 0's then result bit is 0. Otherwise result bit is 1.

Eg: 25 | 15 = 31

$$25 \mid 15 = 31$$

25 = 0000 0000 0001 100 1

15 = 0000 0000 . 0000 1111

0000 0000 0001 1111

$$2^4 + 2^3 + 2^2 + 2^1 + 2^0$$
$$16 + 8 + 4 + 2 + 1 = 31$$

^ - XOR [Exclusive OR]: In this both bits are same then result bit is 0. Otherwise result bit is 1.

Eg: $25 \wedge 15 = 22$

$$25 \wedge 15 = 22$$

25 = 0000 0000 0001 1001

15 = 0000 0000 0000 1111

\wedge

0000 0000 0001 0110

$$\begin{array}{c} 2^4 + 2^2 + 2^1 \\ 16 + 4 + 2 = 22 \end{array}$$

~ - Compliment operator: In compliment operation the bits are complimented. i.e. 1's become 0's and 0's become 1's. Due to this +Ve no becomes -Ve and -Ve no becomes +Ve.

eg: ~25 → -26

$$\begin{array}{rcl} 25 & = & \begin{array}{|cc|cc} 0000 & 0000 & 0001 & 1001 \\ 1111 & 1111 & 1110 & 0110 \end{array} \\ & & \begin{array}{c} \text{ } \quad \quad \quad \diagup \quad \diagdown \quad \quad \diagup \quad \diagdown \\ -128 + 64 + 32 + 4 + 2 = -26 \\ -128 + 102 = -26 \end{array} \end{array}$$

$$25 = 0000\ 0000\ 0001\ 1001$$

$$\sim = 1111\ 1111\ 1110\ 0110$$

$$\begin{array}{c} \diagdown \quad \diagup \quad \diagup \\ 5 \quad 2 \quad 1 \end{array}$$

$$2+4+32+64+128+256+512+1024+2048+4096+8192+16384-32768=-26$$

$$\sim -25 = 0000\ 0000\ 0001\ 1001$$

$$1's\ \sim = 1111\ 1111\ 1110\ 0110$$

$$2's\ \sim = \begin{array}{r} 0000\ 0000\ 0000\ 0001 \\ 1111\ 1111\ 1110\ 0111 \end{array}$$

$$\begin{array}{c} \diagup \quad \diagdown \\ 24 \quad 2 \\ 16+8=24 \end{array}$$

$$\begin{array}{cccc} 1 & 0 & 0 & 01 \\ 0 & 1 & 0 & 1 \\ \hline 1 & 1 & 0 & 10 \end{array}$$

Note: When starting bit is 1 given no is –Ve.

Eg: $\sim -25 \rightarrow +24$

$$\sim -25 = +24$$

$$25 = \begin{array}{|c|c|} \hline 0000 & 0000 \\ \hline \end{array} \quad 0001\ 1001$$

$$\begin{array}{|c|c|} \hline 1111 & 1111 \\ \hline \end{array} \quad 1110\ 0110$$

<== 1's compliment

$$+1$$

<== 2's Complement

$$\begin{array}{cccc} 1111 & 1111 & 1110 & 0111 \\ \sim \rightarrow 0000 & 0000 & 0001 & 1000 \end{array}$$

$$\begin{array}{c} \downarrow \quad \downarrow \end{array}$$

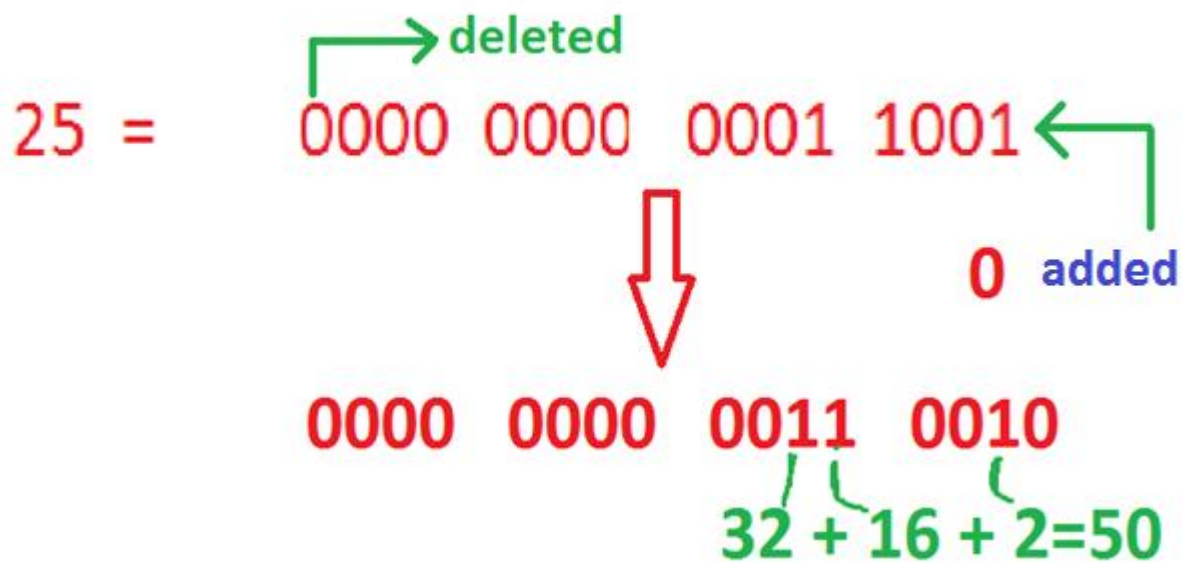
$$16+8=24$$

<< - left shift operator:

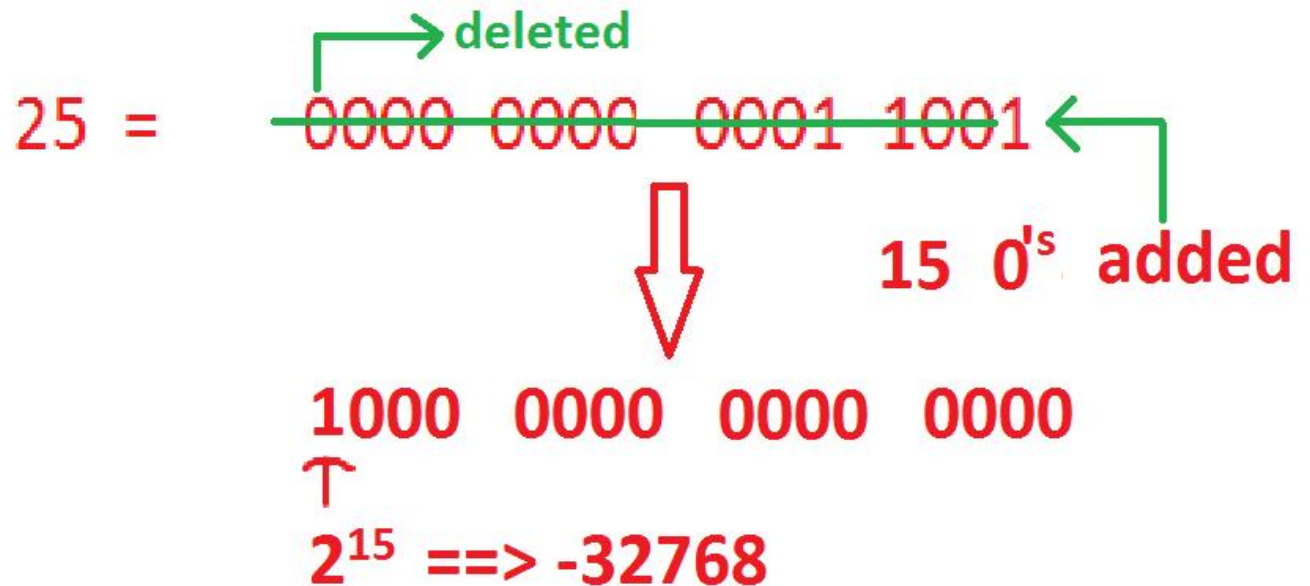
In left shift operation, the specified no of bits are deleted from left side and the same no of **zeros** added on right side. In left shift operation, most probably the value is multiplied with 2 that no of times.

Eg: $25 \ll 1 = 50$, $25 \ll 2 = 100$, $25 \ll 15 = -32768$,
 $25 \ll 16 = 0$

eg: $25 \ll 1 = 50$



eg: $25 \ll 15 = -32768$



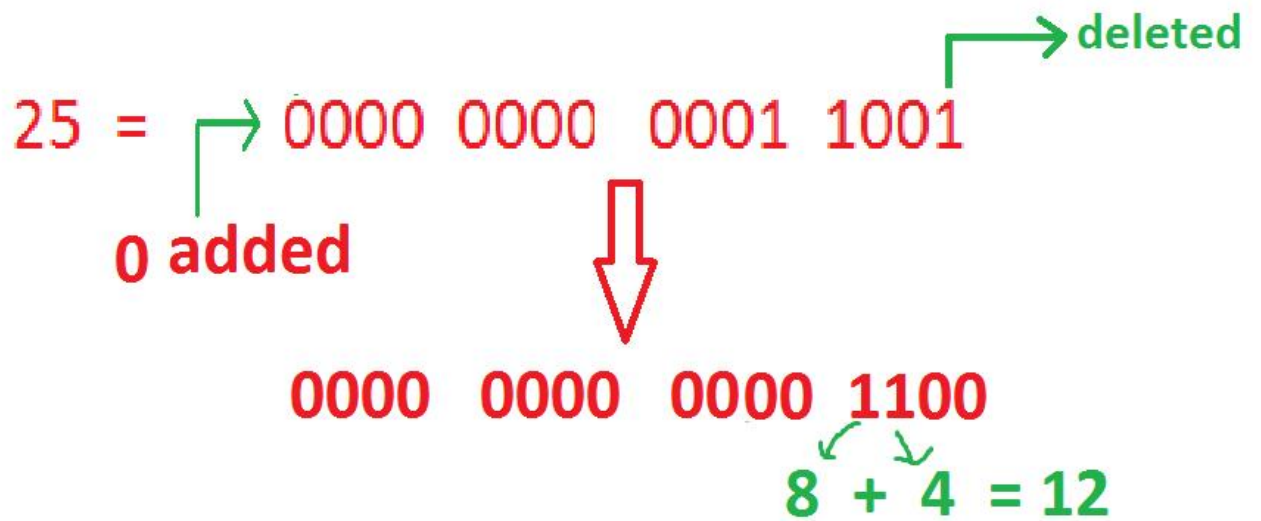
Note: When starting bit 1 no is negative.

>> - Right shift operator:

In right shift operation, the bits are moved to right side i.e. the specified no.of bits are deleted from right side and same no.of **zero's** are added left side. Due to this always the number is divided with 2 that no of times.

Eg: $25 \gg 1 = 12$, $25 \gg 2 = 6$, $25 \gg 3 = 3$, $25 \gg 4 = 1$, $25 \gg 5 = 0$

eg: $25 \gg 1 = 12$



eg: $25 \gg 5 = 0$

