Programming for Problem Solving

2ES104

Exercise-2.1

Demonstrate the use of following operators by a separate program for each

(i) Arithmetic Operators: +, -, *, /, %, ++, --

(ii) Relational Operators: ==, !=, <, >, <=, >=

(iii) Logical operators: &&, ||, !

(iv) Conditional operator: <expr1>?<expr2>:<expr3>

(v) Shorthand assignment: =+, -=, *=, /=, %/

(vi) Increment-Decrement operator:

	Prefix	**Postfix**
Increment	++a;	a++;
Decrement	--a;	a--;

(vii) Bitwise operator: &, |, ^, <<, >> ,~

Arithmetic operator

Operators	Meaning	Example	Result
+	Addition	4+2	6
-	Subtraction	4-2	2
*	Multiplication	4*2	8
/	Division	4/2	2
%	Modulus operator to get remainder in integer division	5%2	1
+ +	Increment	A = 10; A+ +	11
− −	Decrement	A = 10; A− −	9

Relational Operator

Operators	Meaning	Example	Result
<	Less than	5<2	False
>	Greater than	5>2	True
<=	Less than or equal to	5<=2	False
>=	Greater than or equal to	5>=2	True
==	Equal to	5==2	False
! =	Not equal to	5! =2	True

Relational Operator

```c
#include <stdio.h>

int main()
{
	int a = 9;
	int b = 4;

	printf(" a >  b: %d \n", a > b);
	printf(" a >= b: %d \n", a >= b);
	printf(" a <= b: %d \n", a <= b);
	printf(" a <  b: %d \n", a < b);
	printf(" a == b: %d \n", a == b);
	printf(" a != b: %d \n", a != b);

	return 0;
}
```

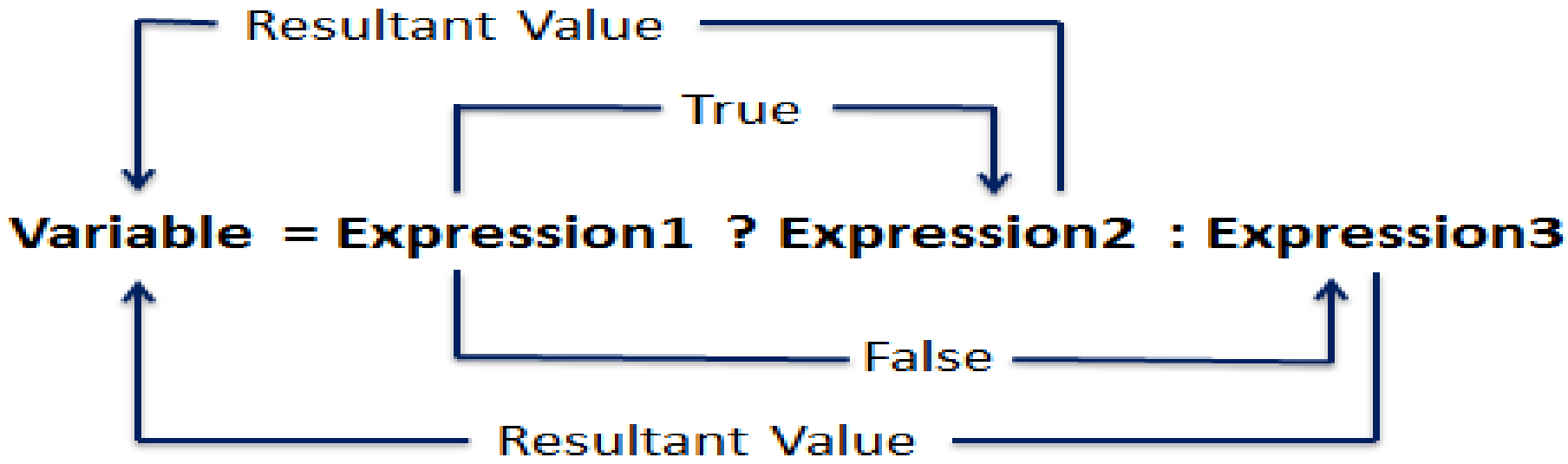
Logical Operator

For all examples below consider a = 10 and b = 5		
Operator	**Description**	**Example**
&&	Logical AND	(a>b) && (b==5) gives true
||	Logical OR	(a>b) || (b==2) gives true
!	Logical NOT	!(b==5) gives false

Logical Operator

```
#include<stdio.h>
   void main()
   {
         int a, b;
        printf("Enter values for a and b : ");
        scanf("%d %d", &a, &b);
        printf("\n %d",(a<b)&&(a!=b));
        printf("\n %d",(a<b)||(b<a));
        printf("\n %d",!(a==b));
   }
```

Conditional Operator




```
void main()
{
int a = 10;
int b = 20;
(a>b) ? printf("a is greater") : printf("b is greater");
}
```

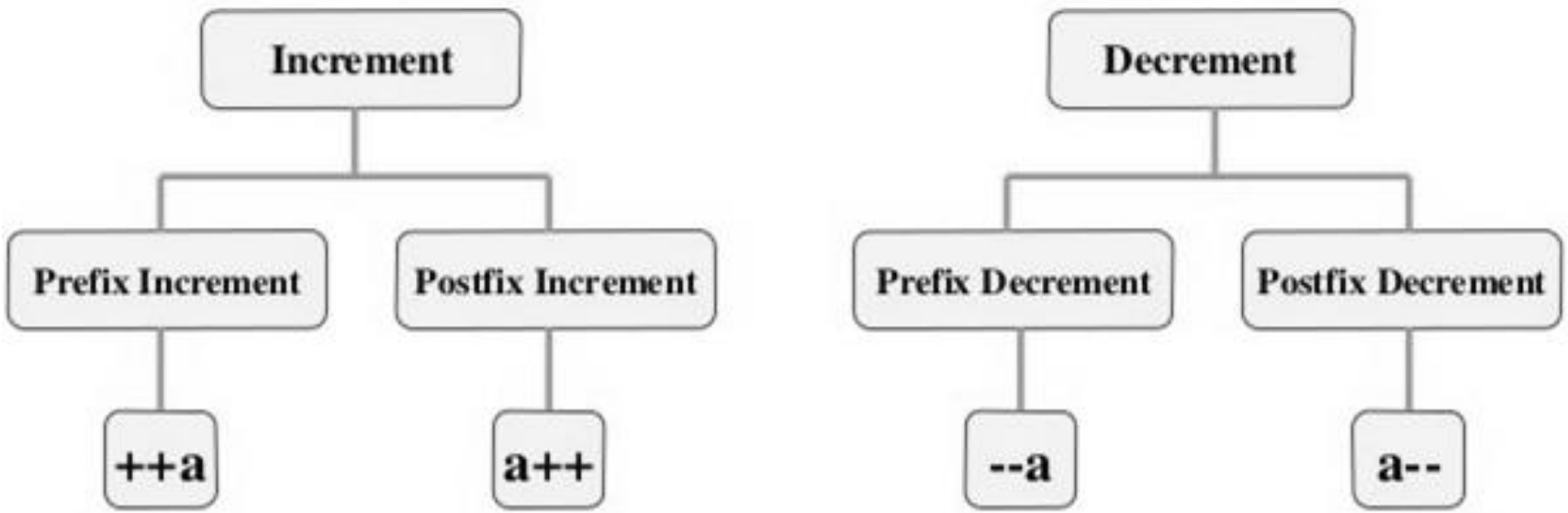
Conditional Operator

```
#include<stdio.h>
void main()
{
int a,b,big;
printf(“Enter two numbers : “);
scanf(“%d %d”,&a,&b); //A=8  B=7
big = (a>b) ? a : b;
printf(“Biggest Number is : %d ”,big);
}
```

Assignment Operator

Operator	Example	Equivalent Expression (m=15)	Result
+=	m +=10	m = m+10	25
-=	m -=10	m = m-10	5
*=	m *=10	m = m*10	150
/=	m /=	m = m/10	1
%=	m %=10	m = m%10	5

Increment and Decrement Operator




```
x = 5
x = x+1
x = 6
```

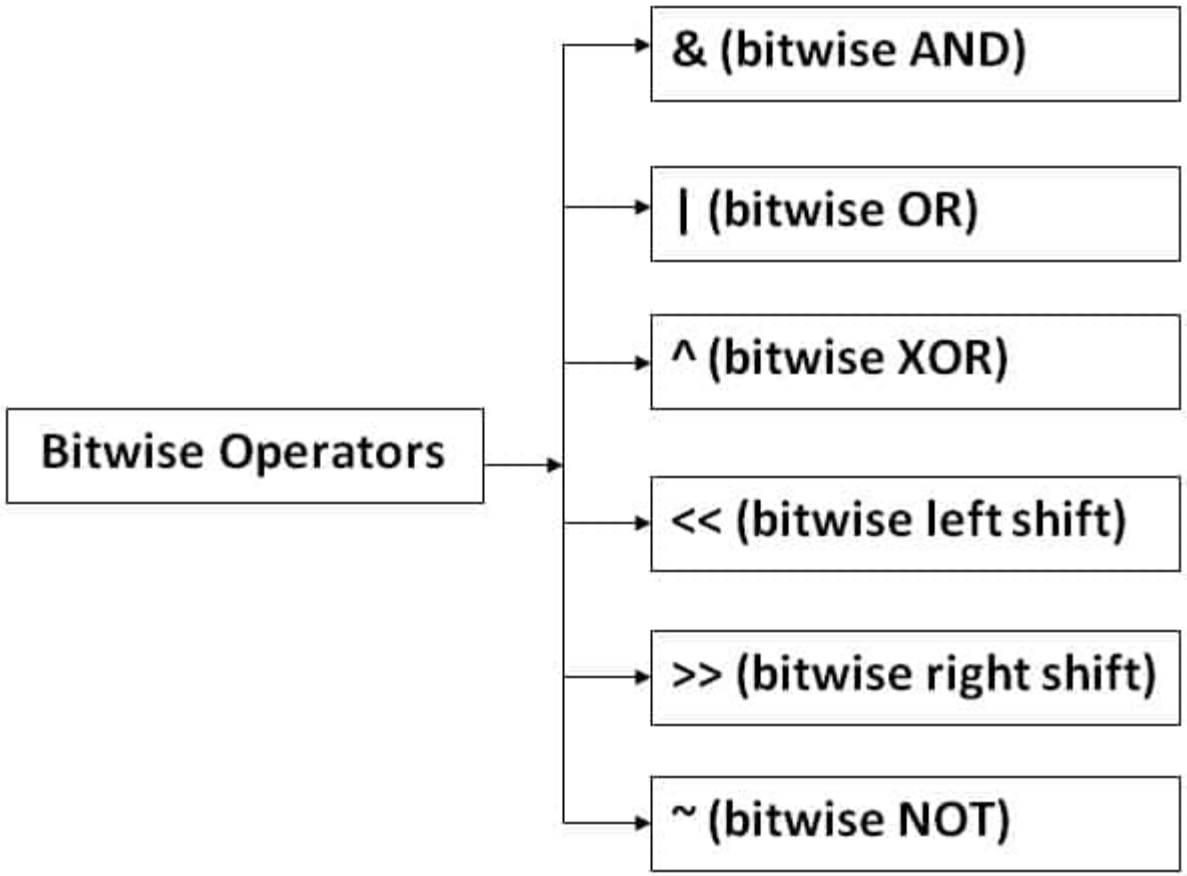
```
x = 5
++x
x = 6
```

```
x = 5
x++
x=5
```

Increment and Decrement Operator

prefix	x=10, y=0	postfix	x=10,y=0
++x	y=++x y=11, x=11	x++	y=x++ Y=10, x=11
--x	y=--x y=9, x=9	x--	y=x-- y=10, x=9

Bitwise Operator



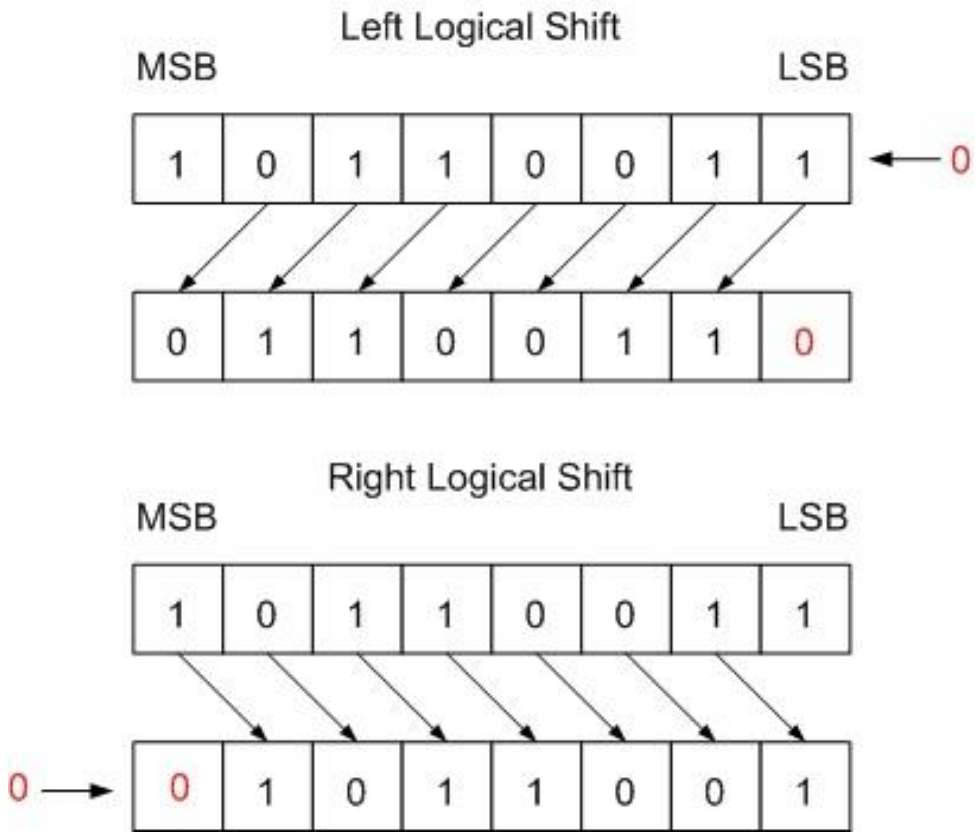
Bitwise Operator

x	y	x|y	x&y	x^y
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

Bitwise Operator

	7	=	0	1	1	1	
	4	=	0	1	0	0	
7 AND	4	=	0	1	0	0	= 4
	7	=	0	1	1	1	
	4	=	0	1	0	0	
7 OR	4	=	0	1	1	1	= 7
	7	=	0	1	1	1	
	4	=	0	1	0	0	
7 XOR	4	=	0	0	1	1	= 3

Bitwise Operator



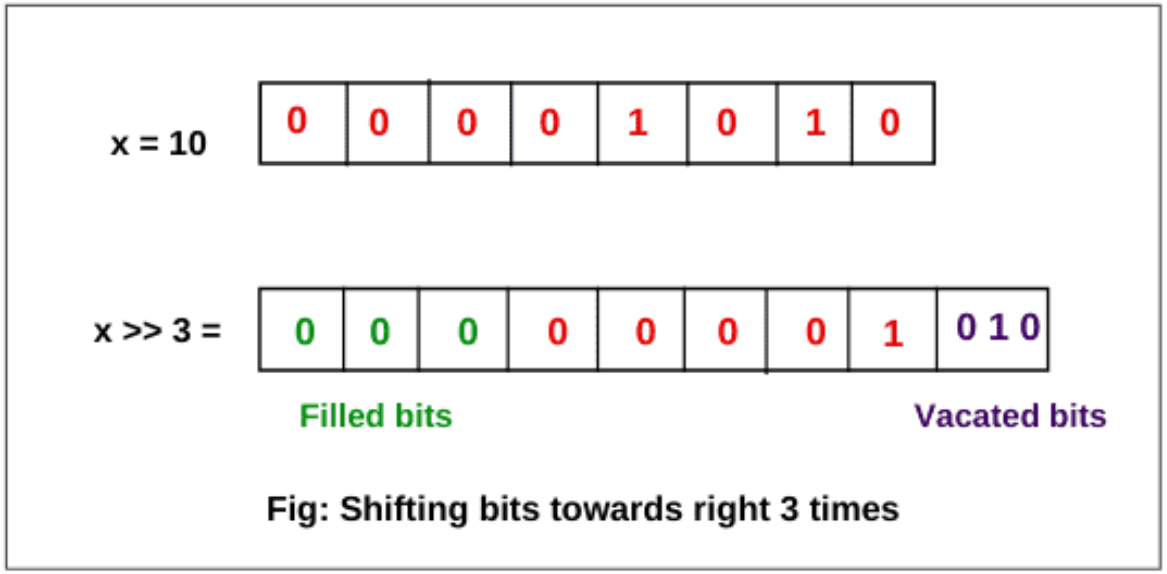


Fig: Shifting bits towards right 3 times

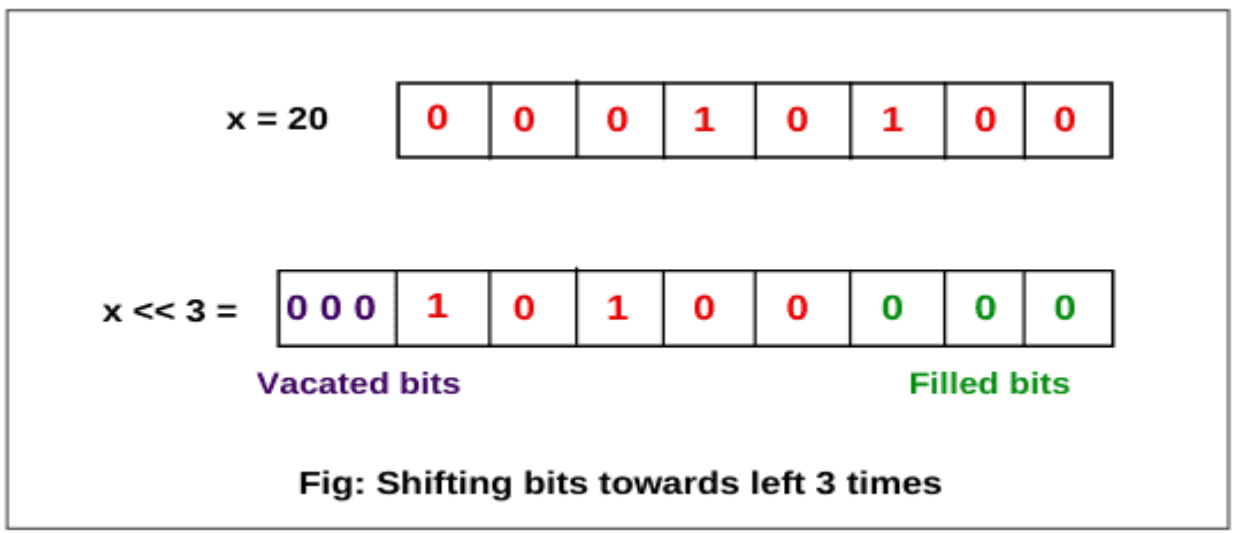


Fig: Shifting bits towards left 3 times

Decimal to Binary

Base^{Exponent}	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Place Value	128	64	32	16	8	4	2	1

Example: Convert decimal 35 to binary	0	0	1	0	0	0	1	1

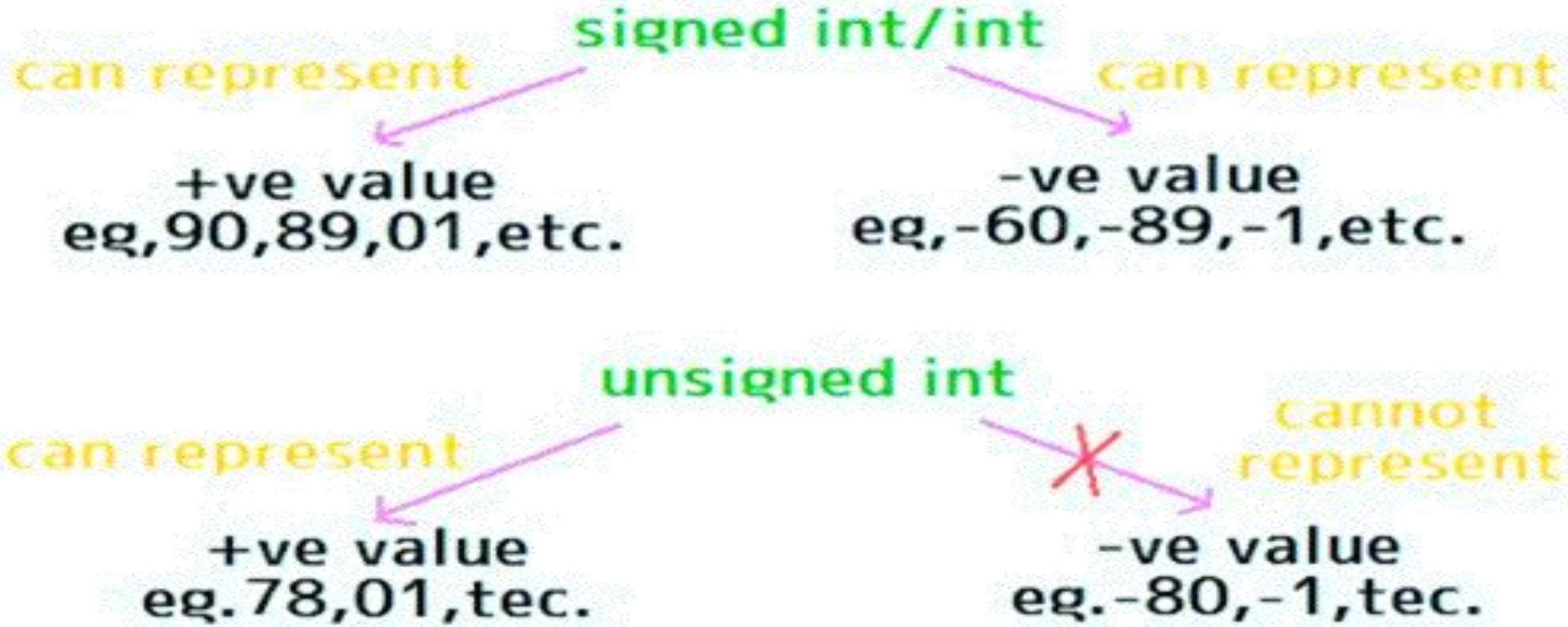
Binary of No :70 = 0 1 0 0 0 1 1 0

Exercise-2.1

(viii) print following table as output using sizeof() operator:

Data type	Format Specifier	Size	Range
Signed char	%c	1 Byte	-128 to 127
Unsigned char	%c	1 Byte	0-255
int, long int or signed int	%d	4 Byte	-2147483648 to 2147483647
Unsigned int or unsigned long int	%u	4 Byte	0 to 4,294,967,295
Short int	%hd	2 Byte	-32768 to 32767
Unsigned short int	%hu	2 Byte	0 to 65535
Float	%f	4 Byte	3.4E-38 to 3.4E+38
Double	%lf	8 Byte	1.7E-308 to 1.7E+308
Long double	%Lf	10 Byte	3.4E-4932 to 1.1E+4932.

Exercise-2.1


signed int/int
can represent
+ve value
eg,90,89,01,etc.
can represent
-ve value
eg,-60,-89,-1,etc.
unsigned int
can represent
+ve value
eg.78,01,tec.
cannot
represent
-ve value
eg.-80,-1,tec.

Exercise-2.1

Suffix	Decimal constants	Octal or hexadecimal constant
none	int long int long long int	int unsigned int long int unsigned long int long long int unsigned long long int
u or U	unsigned int unsigned long int unsigned long long int	unsigned int unsigned long int unsigned long long int
l or L	long int long long int	long int unsigned long int long long int unsigned long long int
Both u or U and l or L	unsigned long int unsigned long long int	unsigned long int unsigned long long int
ll or LL	long long int	long long int unsigned long long int
Both u or U	unsigned long long int	unsigned long long int

Exercise-2.1

				Range	
Type	Sign	Bytes	Bits	Min	Max
char	signed	1	8	-128	127
char	unsigned	1	8	0	255
byte		1	8	0	255
int (Uno +)	signed	2	16	-32768	32767
short		2	16	-32768	32767
int (Uno +)	unsigned	2	16	0	65535
word		2	16	0	65535
int (Due)	signed	4	32	-2147483648	2147483647
long	signed	4	32	-2147483648	2147483647
int (Due)	unsigned	4	32	0	4294967295
long	unsigned	4	32	0	4294967295
float		4	32	-3.4028235E+38	3.4028235E+38
double (Uno +)		4	32	-3.4028235E+38	3.4028235E+38
double (Due)		8	64	(small)	(BIG)

Size of operator

- used to find size of data type in C Language.
    - printf("%lu\n", sizeof(char));
    - printf("%lu\n", sizeof(int));
    - printf("%lu\n", sizeof(float));
    - printf("%lu", sizeof(double));
- When *sizeof()* is used with the expression, it returns size of the expression
    - int a = 0;
    - float d = 10.21;
    - printf("%lu", sizeof(a + d));