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;
}
```

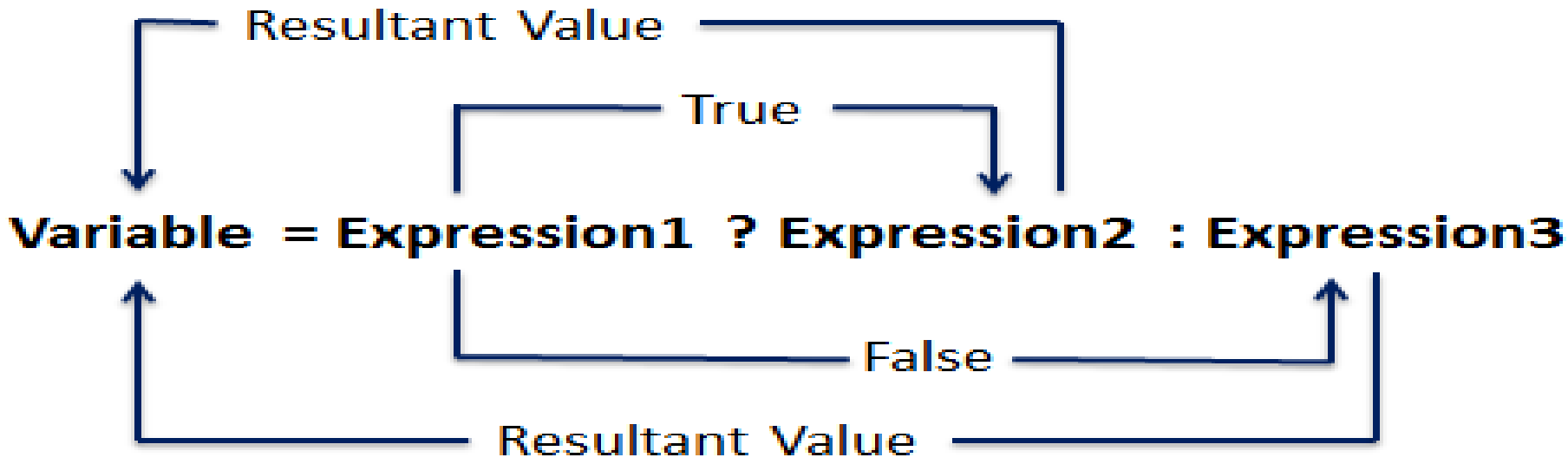
Logical Operator

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");
}
```

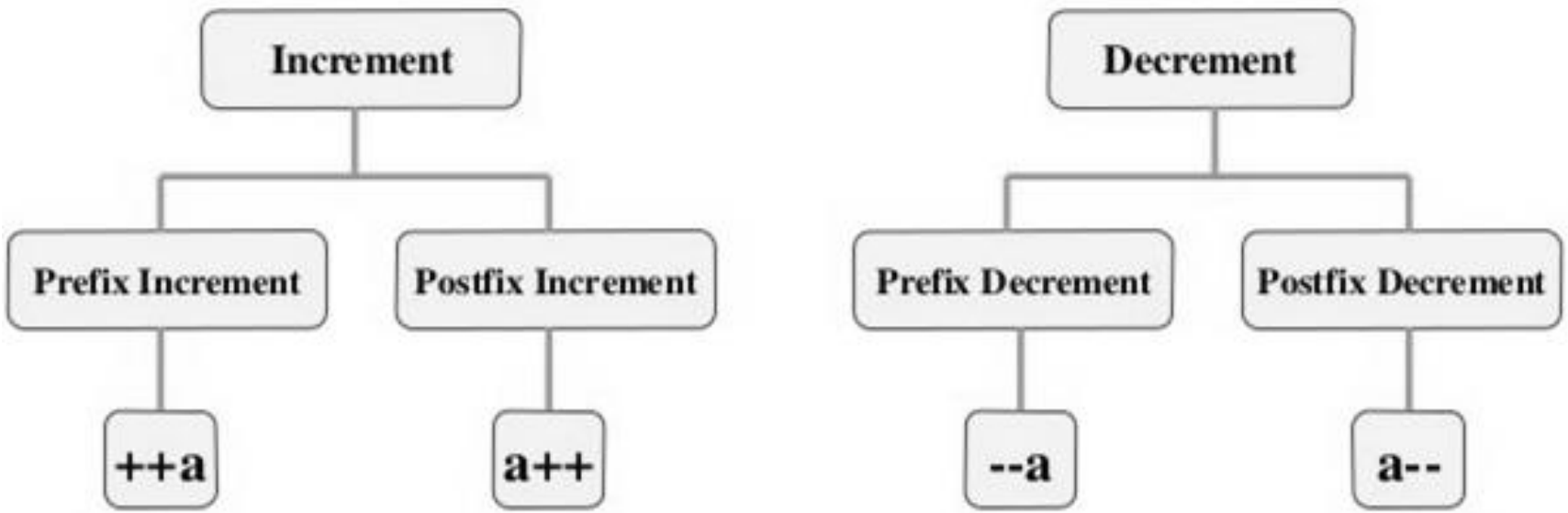
Conditional Operator

```
#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
```

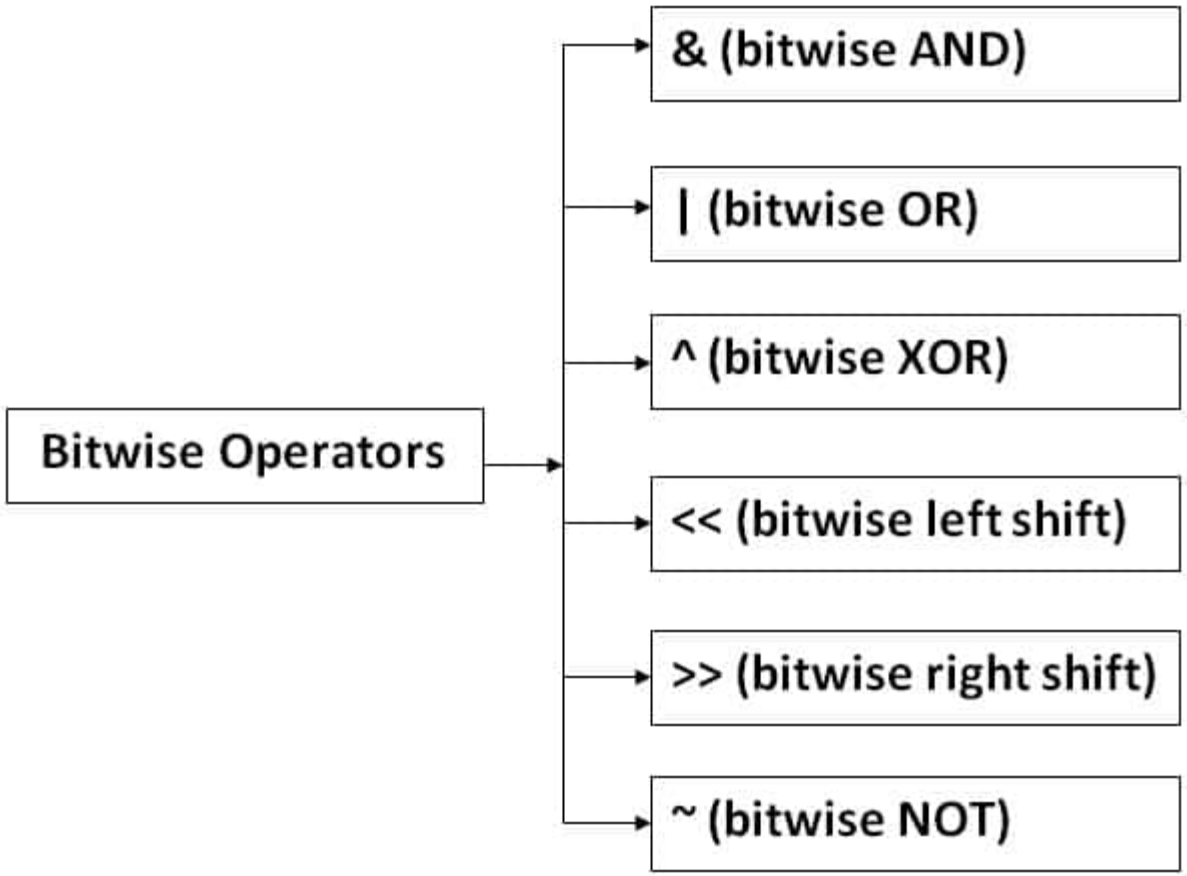
```
x = 5
++x
x = 6
```

```
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



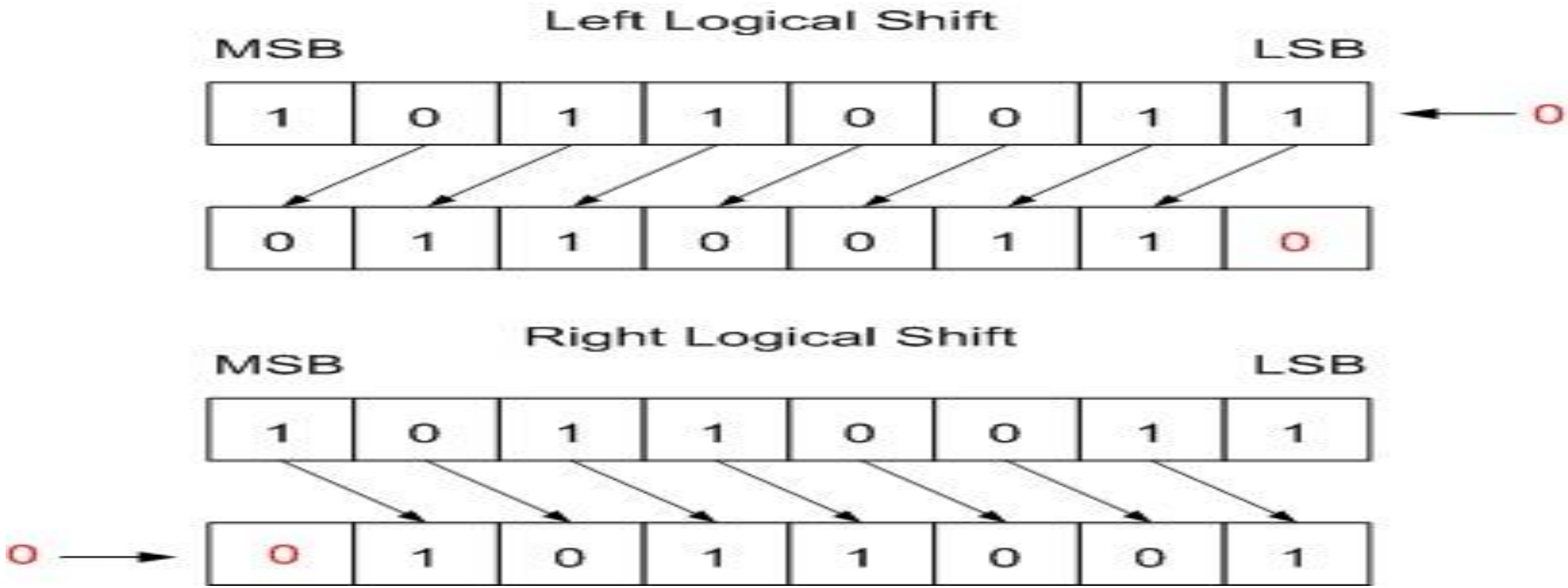
Bitwise Operator

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



Bitwise Operator

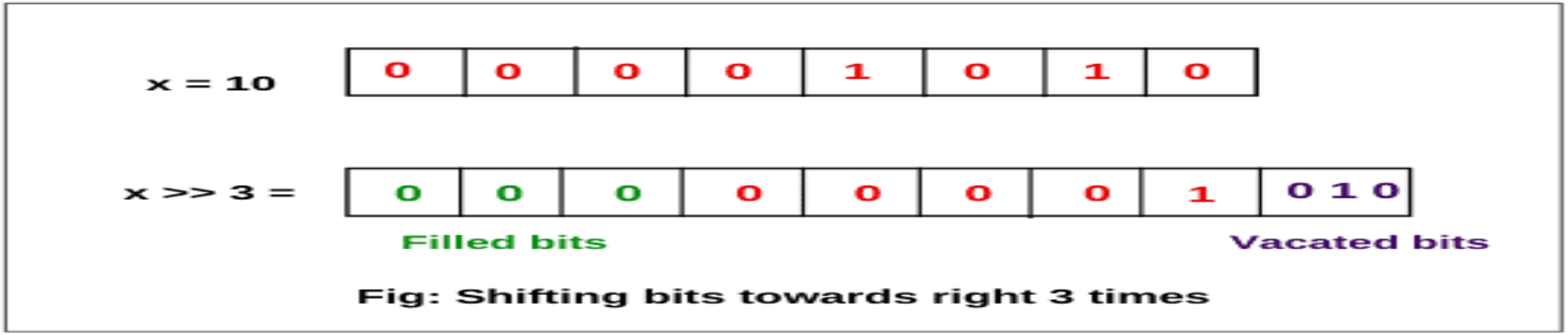


Fig: Shifting bits towards right 3 times

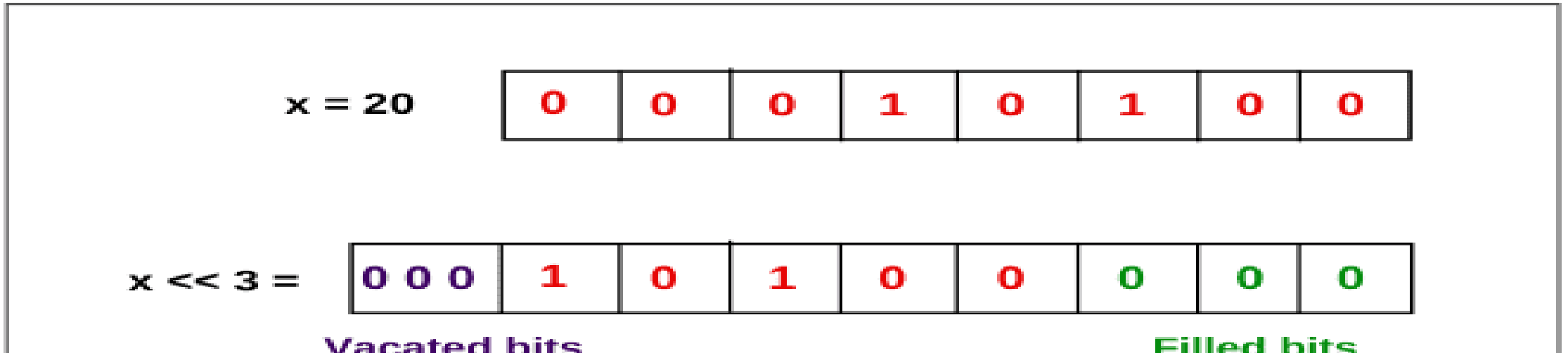


Fig: Shifting bits towards left 3 times

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

1 0 1 1 1 0 0

1

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.

Range calculation

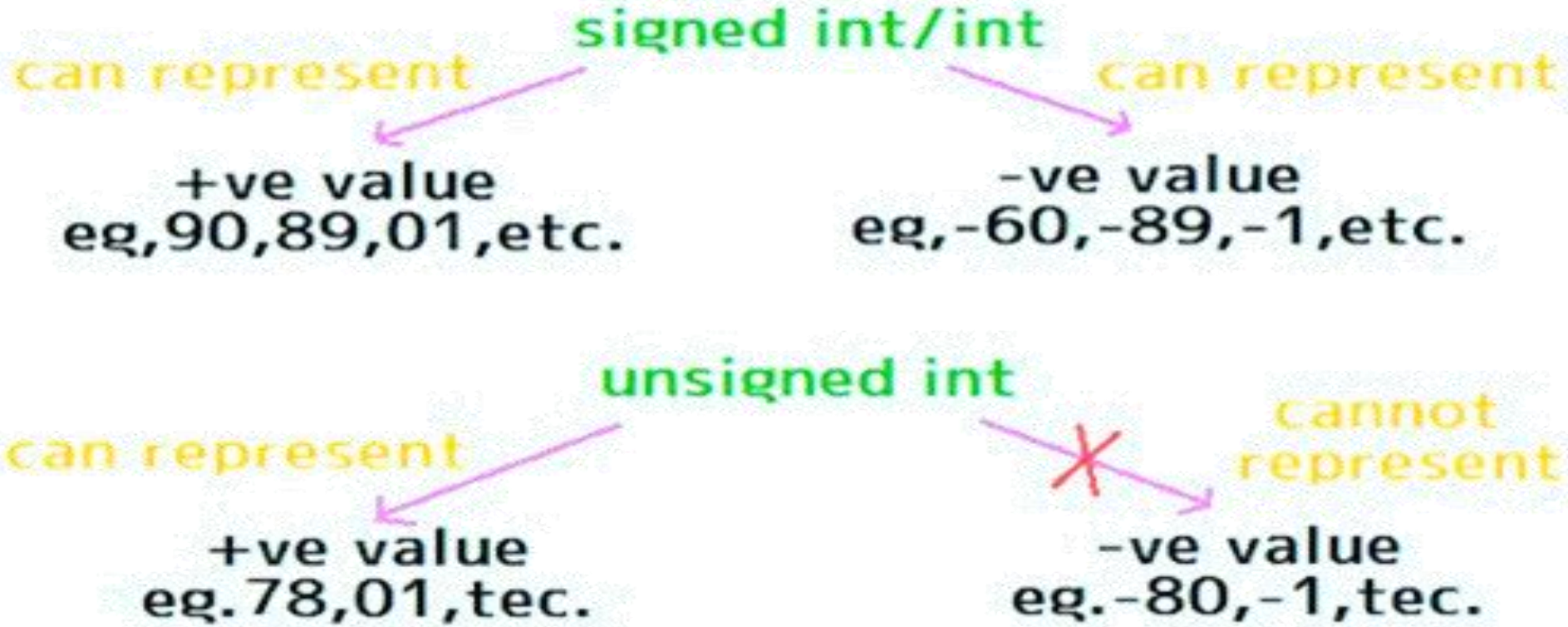
- signed type is given by –

$$-(2^{N-1}) \text{ to } 2^{N-1} - 1$$

- unsigned type is given by –

$$0 \text{ to } (2^{N-1}) + (2^{N-1} - 1)$$

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

				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));

Practical-2.2

- Write a program which prints integer variable (15), and floating variable (20.153) in following formats:

  (a) 15 20.153000 (tab in between)

  (b) 15 20.153000 (space in between)

  (c) 15 is an integer constant and 20.153000 is a floating constant.

  (d) 15

  20.153000

The const keyword

- Variables can be declared as constants by using the “const” keyword before the datatype of the variable.
- The constant variables can be initialized once only.
- The default value of constant variables are zero.
- E.g
  - const int a=10;
  - const float k=40.234;

Practical-2.3

- Write a program to display sum of two variables taken input from keyboard and display its result like
  - The sum of integer variables A and B is: answer
  - The sum of float variables C and D is: answer
  - The sum of character variables X and Y is: answer

ASCII Value

0	NUL	16	DLE	32	SP	48	0	64	@	80	P	96	`	112	p
1	SOH	17	DC1	33	!	49	1	65	A	81	Q	97	a	113	q
2	STX	18	DC2	34	"	50	2	66	B	82	R	98	b	114	r
3	ETX	19	DC3	35	#	51	3	67	C	83	S	99	c	115	s
4	EOT	20	DC4	36	$	52	4	68	D	84	T	100	d	116	t
5	ENQ	21	NAK	37	%	53	5	69	E	85	U	101	e	117	u
6	ACK	22	SYN	38	&	54	6	70	F	86	V	102	f	118	v
7	BEL	23	ETB	39	'	55	7	71	G	87	W	103	g	119	w
8	BS	24	CAN	40	(	56	8	72	H	88	X	104	h	120	x
9	HT	25	EM	41	)	57	9	73	I	89	Y	105	i	121	y
10	LF	26	SUB	42	*	58	:	74	J	90	Z	106	j	122	z
11	VT	27	ESC	43	+	59	;	75	K	91	[	107	k	123	{
12	FF	28	FS	44	,	60	<	76	L	92	\	108	l	124	|
13	CR	29	GS	45	-	61	=	77	M	93	]	109	m	125	}
14	SO	30	RS	46	.	62	>	78	N	94	^	110	n	126	~
15	SI	31	US	47	/	63	?	79	O	95	_	111	o	127	DEL

Example

```
#include<stdio.h>
void main()
        {
        char a,b, charsum;
        printf("\n Enter the value of a and b:\n ");
        scanf("%c %c",&a,&b); // a=!   b="
        charsum =a+b; // 33+34=67
        printf("\n charsum = %d and %c", charsum,charsum); //67 and C
}
```

fflush(stdin) in C

- fflush() is typically used for output stream only.
- Its purpose is to clear (or flush) the output buffer and move the buffered data to console.
- Example

```
#include <stdlib.h>

Void main()   {

Char a,b,c;      fflush(stdin);

Scanf(“%c%c”,&a,&b);

Printf(“%c%c”,a,b);     }
```

Practical-2.4

- Write a Program to convert the Celsius to Fahrenheit.

$$F = (C * t) + T$$

(declare and use t=1.8 as const variable

and T=32 as #define directive)

#define preprocessor directive

- Variables can be declared as constants by using the #define preprocessor directive
- **#define is** used to **define** some values **with a** name(string)
- It replaces into the code with its value before compilation.
- **const is** a keyword or used to make the value of an identifier constant
- By declaring this variable, it occupies memory unit.
- But we cannot update the value of constant type variable directly

Practical-2.5

- Write a Program to find out simple interest.

  Interest = (Principle * Rate * No of year) / 100)

- Input : variable P ,R, N

- Output: Interest=Answer

Practical-2.6

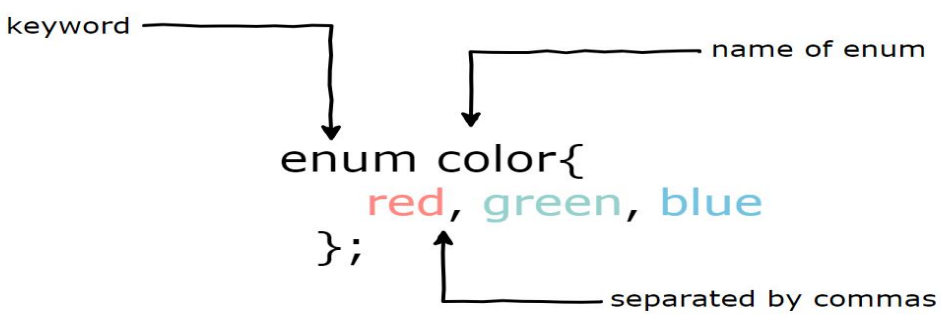
- Write a single C program to demonstrate the use of "typedef" and "enum".

Typedef keyword

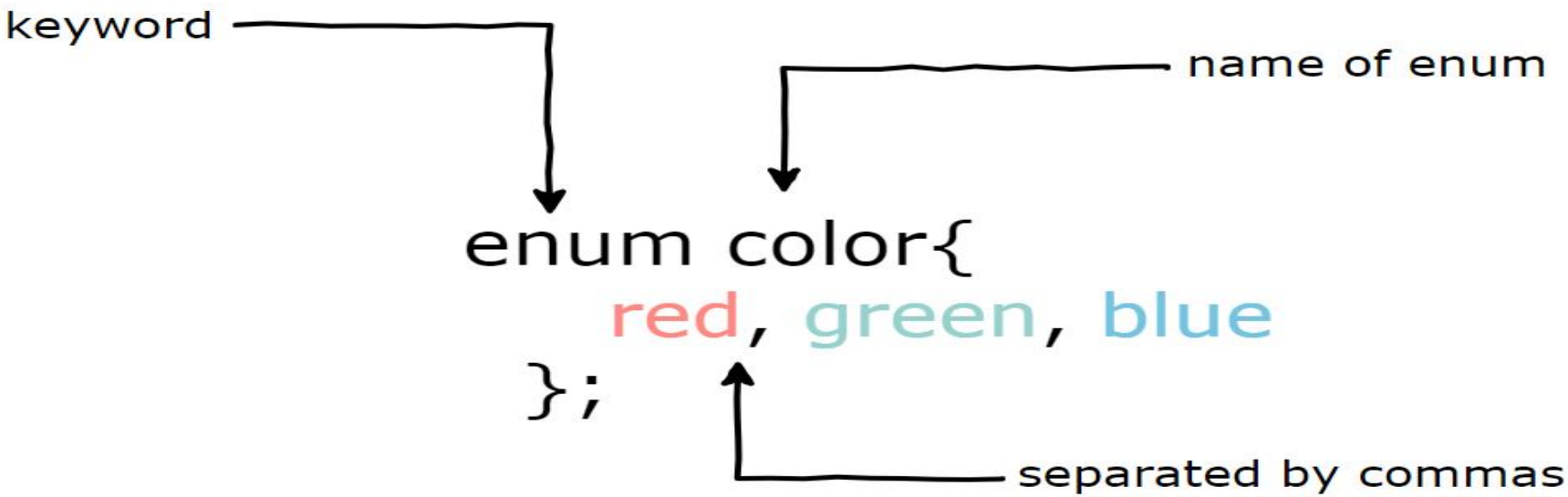
- Used to assign alternative names to existing datatype.
- Its mostly used with user defined datatypes
- Syntax:
  - typedef <existing_name> <alias_name>
  - typedef **unsigned long ulong;**
  - **ulong** i,j

Enum keyword

- The enum in C is also known as the enumerated type.
- It is used to assign names to the integral constants, which makes a program easy to read and maintain.
- The enum keyword is used to create an enum.
- The constants declared inside are separated by commas.
- E.g



Enum keyword



Enum keyword :Example

```
#include<stdio.h>
enum color {   red ,  green,  blue =32  } ;
int main()
{  // Initializing a variable that will hold the enums
   enum color  current_color  = red;
Int a=10
    printf("Value of red = %d \n", current_color);
    current_color= blue;
    printf("Value of red = %d \n", current_color);
    return 0;
}
```

Output:

value of red=0
value of blue=32

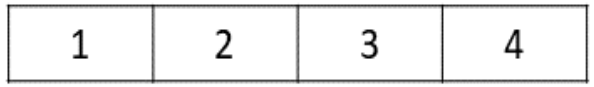
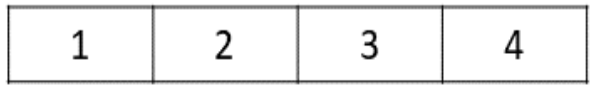
Practical-2.7

- Type and run some sample programs given below and justify their output:

Practical-2.7

Value	Placeholder	Output (□ means blank)
-10	%d	-10
	%2d	-10
	%4d	□-10
	%-4d	-10□
10	%04d	0010
49.76	%.3f	49.760
	%.1f	49.7
	%-10.2f	49.76□□□□□
	%10.2f	□□□□□ 49.76
	%10.3e	□4.976e+01

Practical-2.7

Format	***Output***
printf(“ %d ”,1234);	
printf(“ %7d ”,1234);	
printf(“ %2d ”,1234);	
printf(“ %-7d ”,1234);	
printf(“ %07d ”,1234);	

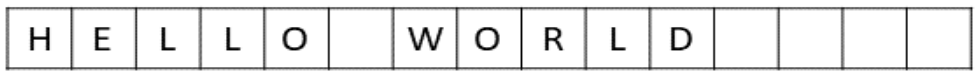
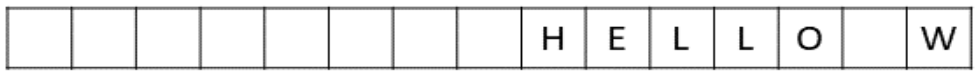
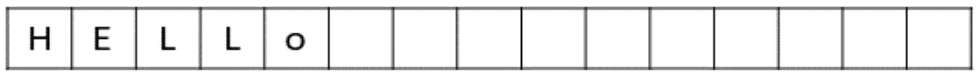
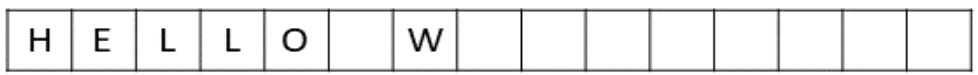
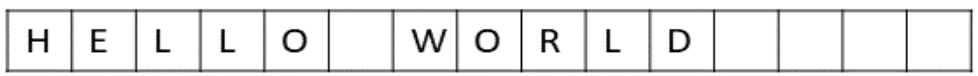
Practical-2.7

Format	Output
printf("%f",y);	25.454500
printf ("%7.4f",y);	25.4545
printf("%7.3f",y);	25.455
printf("%-7.3f",y);	25.455
printf("%10.3e",y);	2.545e+01
printf("%10.3e",-y);	-2.545e+01
printf("%e",y);	2.545450e+01

Practical-2.7

Value	Placeholder	Output (□ means blank)	Interpretation
l='a'	printf("%c",l);	a	
	printf("%3c,l);	□□a	display in 3 columns, right justified
	print("%-3c",l);	a□□	display in 3 columns, left justified
	printf("% *c",4,l); (compiler/machine dependent)	□□□a	display in variable number of columns, specified here as 4
	printf("%*c",-4,l); (compiler/machine dependent)	a□□□	display in variable number of columns, specified here as -4, i.e. left justification

Practical-2.7

Format	***Output***
printf(“ %s ”,”HELLO WORLD”);	
printf(“ %15s ”, ” HELLO WORLD”);	
printf(“ %15.7s ”, ” HELLO WORLD”);	
printf(“ %.5s ”, ” HELLO WORLD”);	
printf(“ %-15.7s ”, ” HELLO WORLD”);	
printf(“ %5s ”, ” HELLO WORLD”);	

Practical-2.7

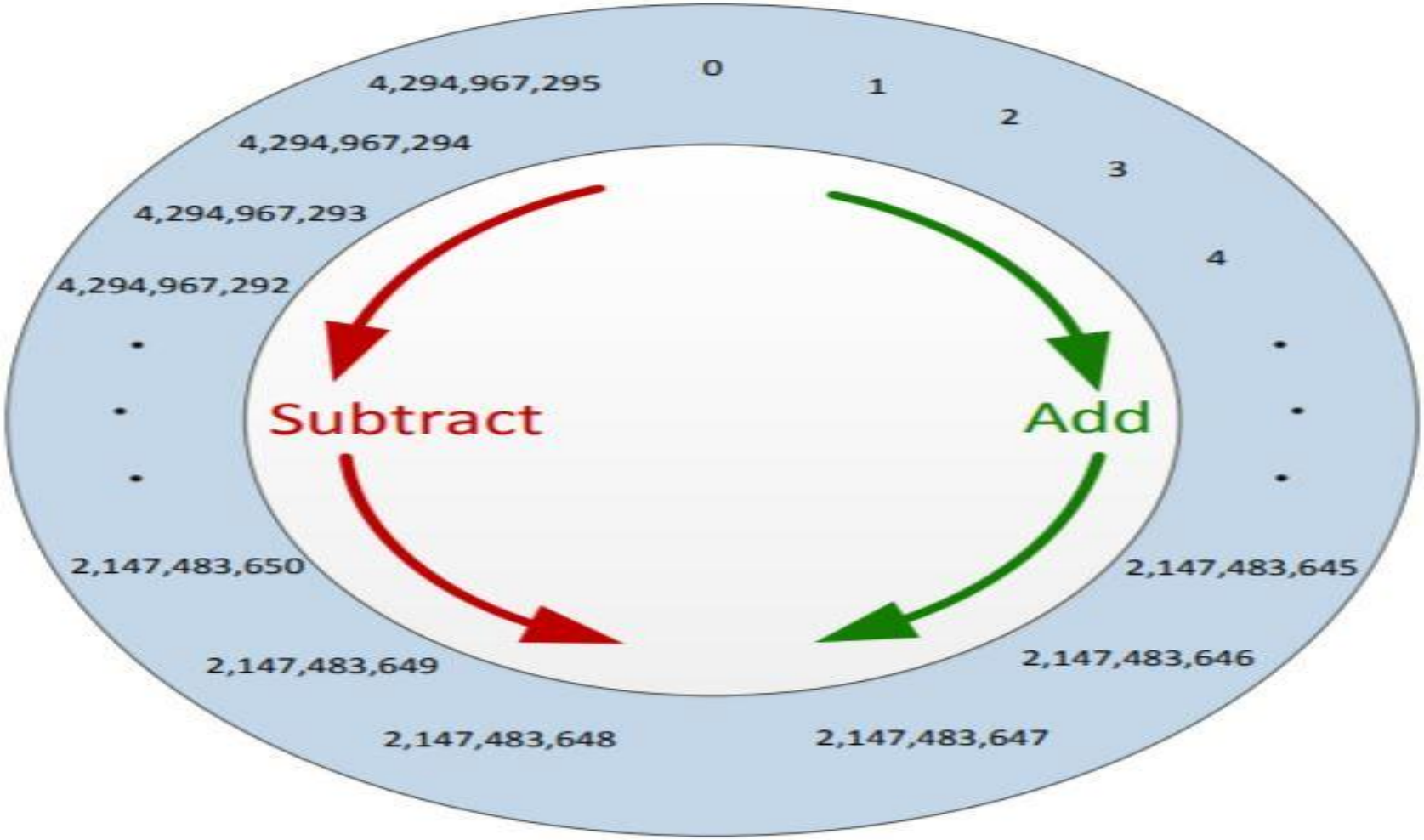
(a)

```
#include<stdio.h>
int main()
{
printf("The color: %s\n",
"blue");
printf("First number: %d\n",
12345);
printf("Second number:
%04d\n", 25);
printf("Third number: %i\n",
1234);
printf("Float number:
%3.2f\n", 3.14159);
printf("Hexadecimal: %x\n",
255);
printf("Octal: %o\n", 255);
printf("Unsigned value: %u\n",
150);
printf("Just print the
percentage sign %%\n", 10);
return 0;
}
```

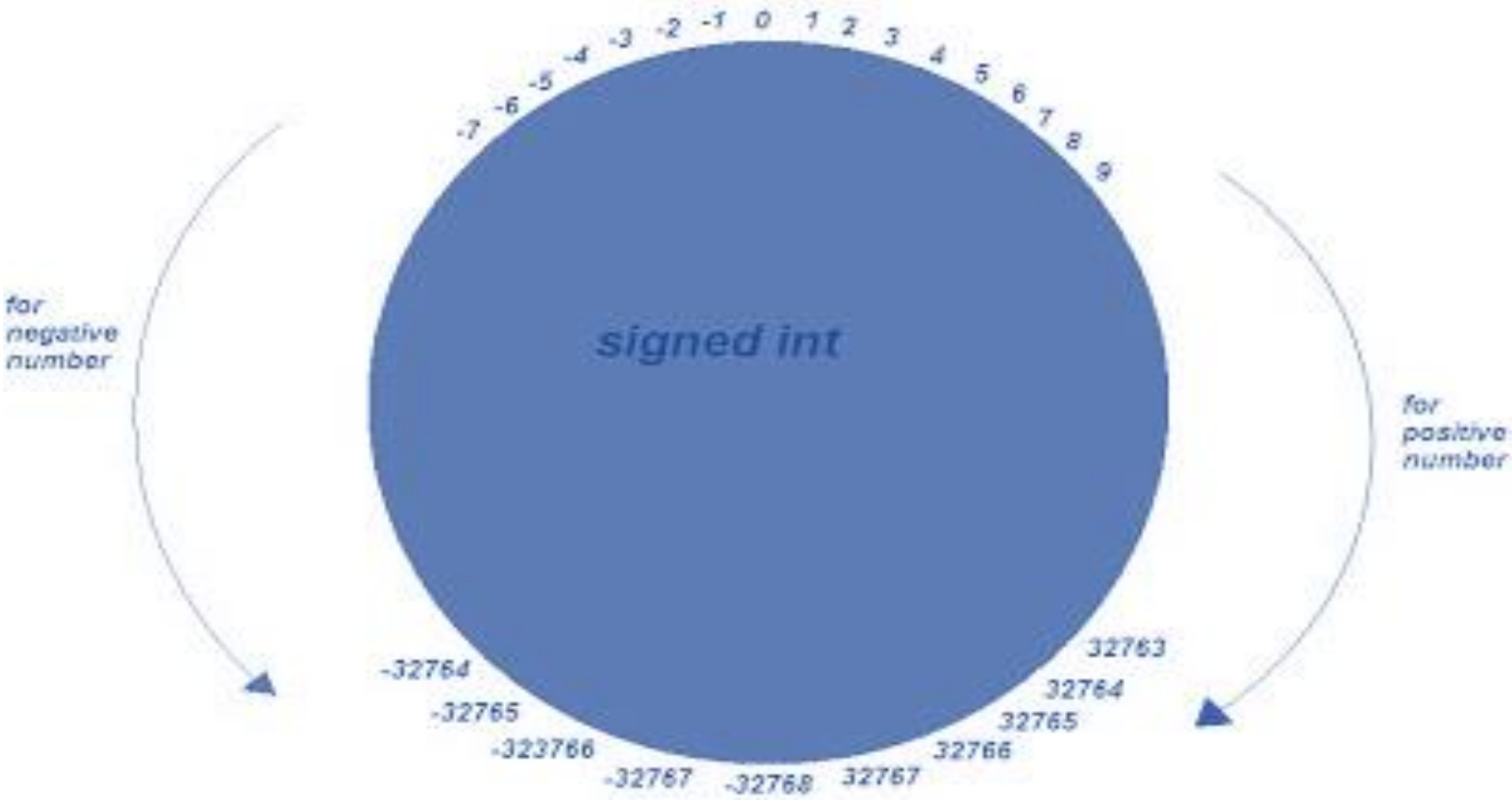
(b)

```
#include<stdio.h>
int main()
{
printf(":%s:\n", "Hello,
world!");
printf(":%15s:\n", "Hello,
world!");
printf(":%.10s:\n", "Hello,
world!");
printf(":%-10s:\n", "Hello,
world!");
printf(":%-15s:\n", "Hello,
world!");
printf(":%.15s:\n", "Hello,
world!");
printf(":%15.10s:\n", "Hello,
world!");
printf(":%-15.10s:\n", "Hello,
world!");
return 0;
}
```

Practical-2.7 (unsigned int)



Practical-2.7 (signed int range: -32768 to 32767)



Practical-2.7

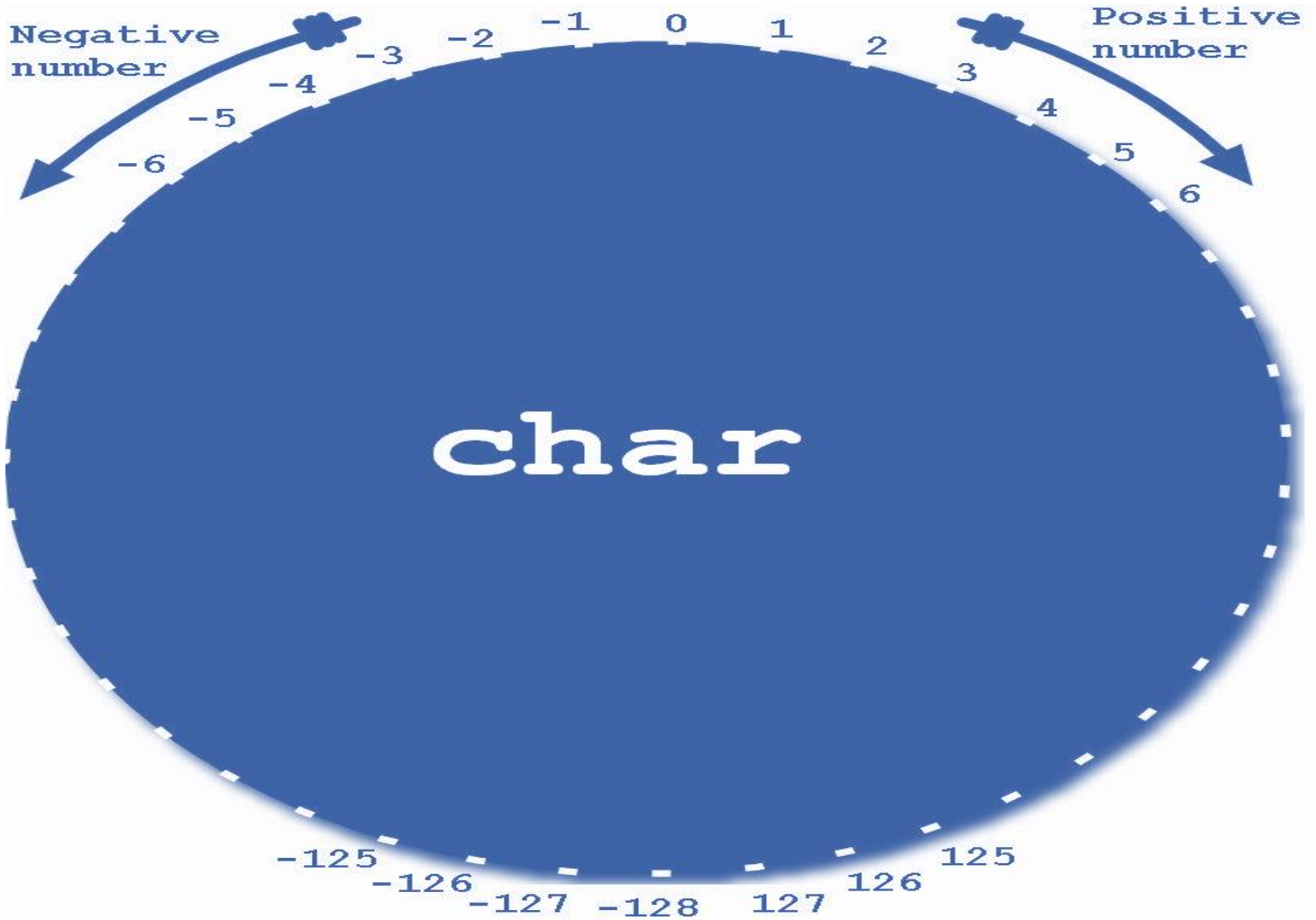
(c)

```
#include<stdio.h>;
void main()
{
int a=5,b=2147483647;
int c=2147483648;
int d=2147483649;
int e=-2147483648,f=-
2147483649,g=-2147483650;
printf("%d %u\n",a,a);
printf("%d %u\n",b,b);
printf("%d %u\n",c,c);
printf("%d %u\n",d,d);
printf("%d %u\n",e,e);
printf("%d %u\n",f,f);
printf("%d %u\n",g,g);
}
```

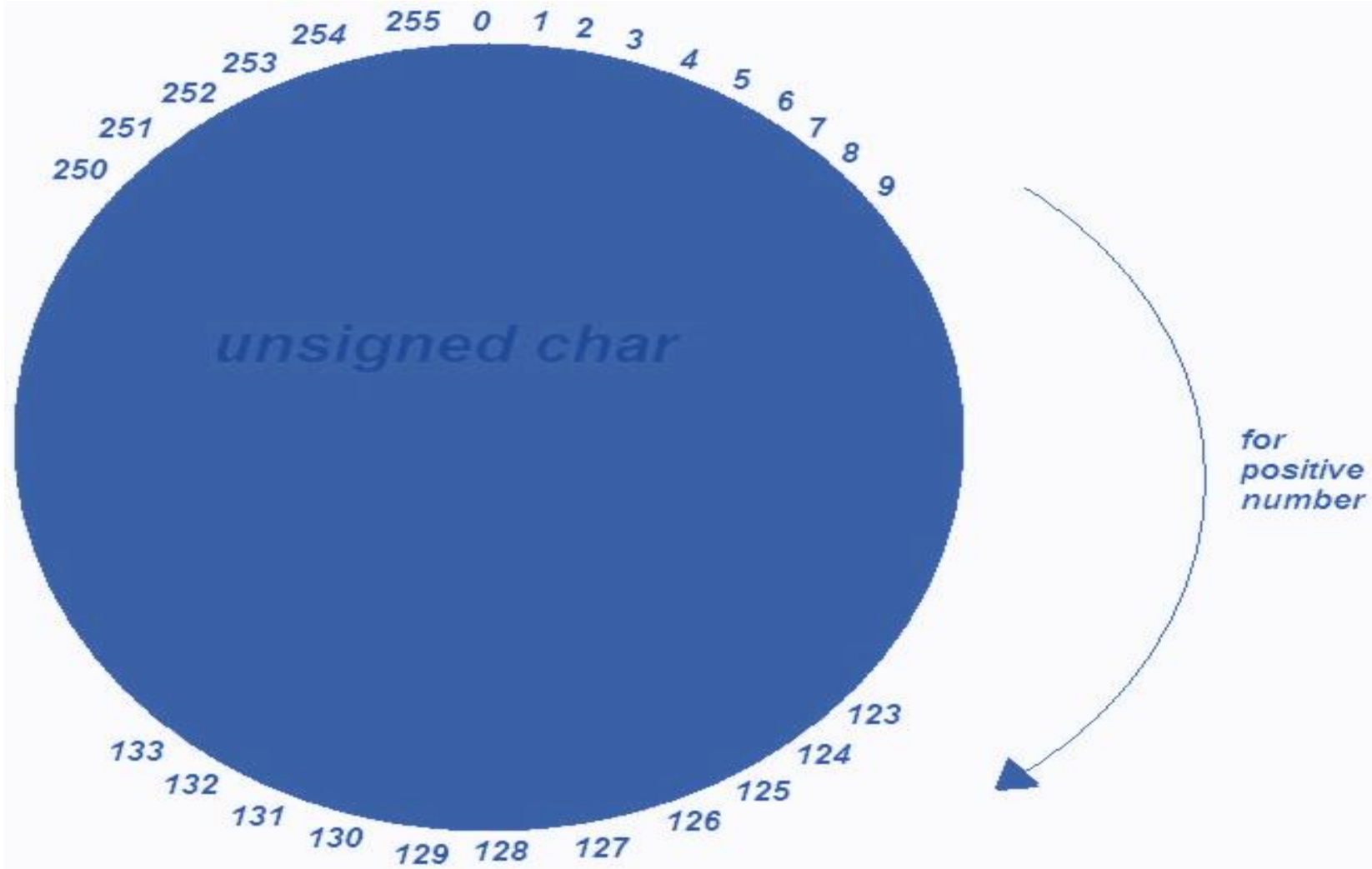
(d)

```
void main()
{
unsigned int
a=2147483647,b=2147483648;
unsigned int c=4294967295;
unsigned int d=4294967296;
unsigned int e=4294967297,f=-
1,g=-2;
printf("%d %u\n",a,a);
printf("%d %u\n",b,b);
printf("%d %u\n",c,c);
printf("%d %u\n",d,d);
printf("%d %u\n",e,e);
printf("%d %u\n",f,f);
printf("%d %u\n",g,g);
}
```

Practical-2.7 (signed char)



Practical-2.7 (unsigned char)



Practical-2.7

Format	Name	Precision	(int)1234	(double)123.456
C or c	Currency	Decimal Places	C = £1,234.00 C4 = £1,234.0000	C = £1,234.56 C4 = £1,234.560
D or d	Decimal	Minimum Digits	D = 1234 D6 = 001234	n/a
E or e	Exponential	Decimal Places	e = 1.230000e+002 E3 = 1.234E+002	e = 1.234567e+002 E3 = 1.235E+002
F or f	Fixed Point	Decimal Places	F = 1234.00 F4 = 1234.0000	F = 1234.57 F4 = 1234.5670
G or g	General	Significant Digits	G = 1234 G2 = 12E+03	G = 1234.567 G4 = 1235
N or n	Number	Desired Decimal Places	N = 1,234.00 N4 = 1,234.0000	N = 1,234.57 N4 = 1,234.5670
P or p	Percent	Decimal Places	P = 123,400.00 % P0 = 123,400 %	P = 123,456.70 % P0 = 123,457 %
R or r	Round-Trip	n/a	n/a	R = 1234.567
X or x	Hexadecimal	Number of Digits	x = 4d2 X4 = 04D2	n/a

Practical-2.7

(e)

```
#include<stdio.h>
void main()
{
char j='xyz'; //you will get
an error.
char a='p', b='pr', c=97,
d=48, e=305,f=97;
char g='a';

char h='125'; //you will get
an error.
char i;
printf("%c %d\n",a,a);
printf("%c %d\n",b,b);
printf("%c %d\n",c,c);
printf("%c %d\n",d,d);
printf("%c %d\n",e,e);
printf("%c %d\n",f,f);
printf("%c %d\n",g,g);
printf("%c %d\n",h,h);
}
```

(f)

```
#include<stdio.h>
void main()
{
long double d = 3.1415926535;
short int i = 3;
long int j = 3;
printf("We have %d cats\n",3);
printf("%g\n", d);
printf("%Lg\n", d);
printf("%g\n", 93000000.0);
printf("%g\n", d);
printf("%hd\n", i);
printf("%ld\n", j);
printf("%.3f\n", 1.2);
printf("%.3f\n", 1.2348);
printf("%.3f\n%.3g\n%.3f\n%.3g\
n", 100.2, 100.2, 3.1415926,
3.1415926);
}
```

Format specifier x

```
#include <stdio.h>
int main()
 {
     int a = 100;
     printf("%010x and %X  and %#x \n", a,a,a);

 return 0;
  }
```

OUTPUT:

0000000064 and 64 and 0x64

Practical-2.7

(g)

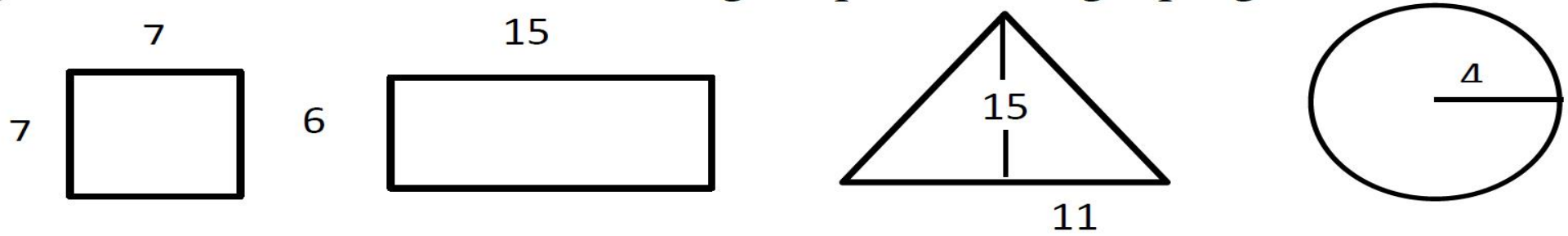
```
#include<stdio.h>
void main()
{
printf("%.0d", 0);
printf("%.3d", 10);
printf("%.5s\n", "abcdefg");
printf("%5s\n", "abc")
printf("%8.5f\n", 1.234);
}
```

(h)

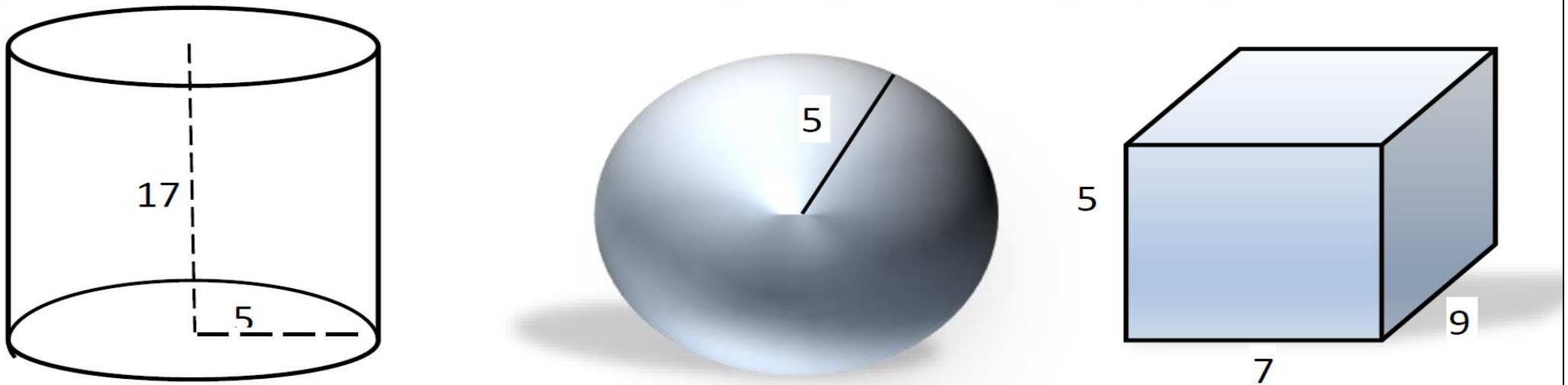
```
#include<stdio.h>
void main()
{
printf("%#x", 12);
printf("%x", 12);
printf("%05d\n", 10);
printf("%+d\n", 10);
printf("|%-5d|%-5d|\n", 1, 2);
printf("%#010x\n", 12);    }
```

Practical-2.8

(i) Calculate the area of following shapes in a single program:

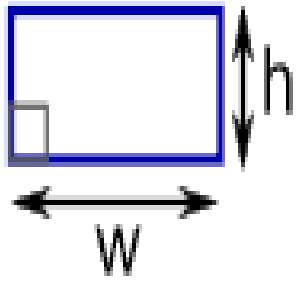


(ii) Calculate the volume of following shapes in a single program:



$V=\pi r^2 h$ $V = \frac{4}{3}\pi r^3$ V= L * W * H

Practical-2.8

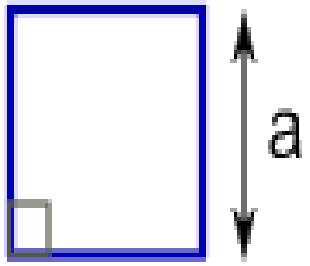


Rectangle

Area = w x h

w = width

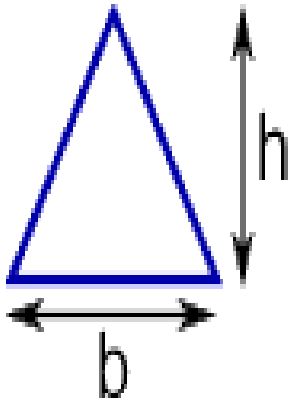
h = height



Square

Area = a^2

a = length of side

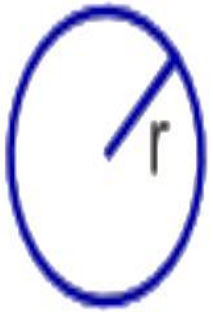


Triangle

Area = ½ x b x h

b = base

h = vertical height



Circle

Area = $\pi \times r^2$

Circumference = $2 \times \pi \times r$

r = radius