Programming for Problem Solving

2ES104

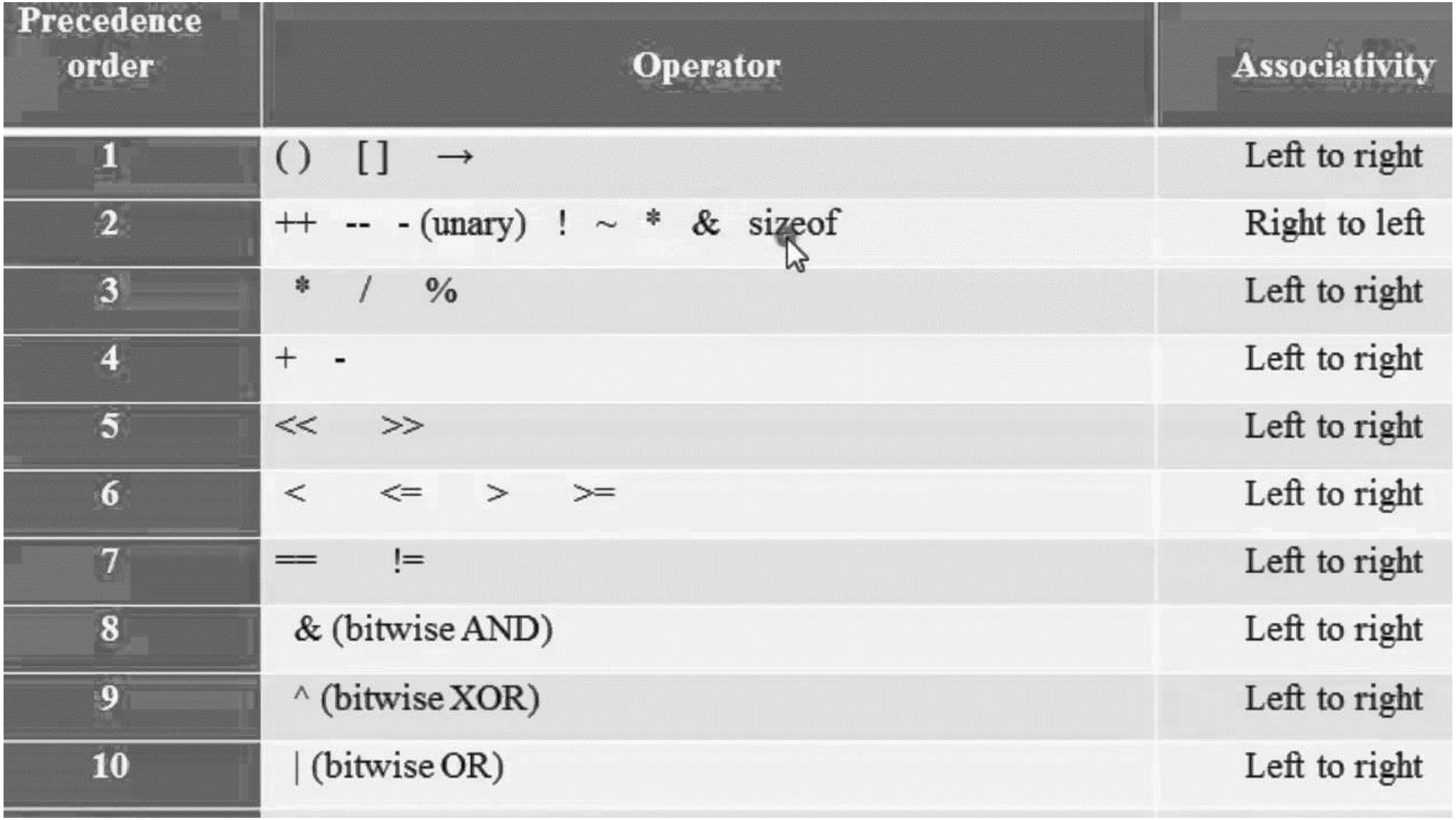
Practical-3.1

Write a program which makes use of following mathematical expressions.

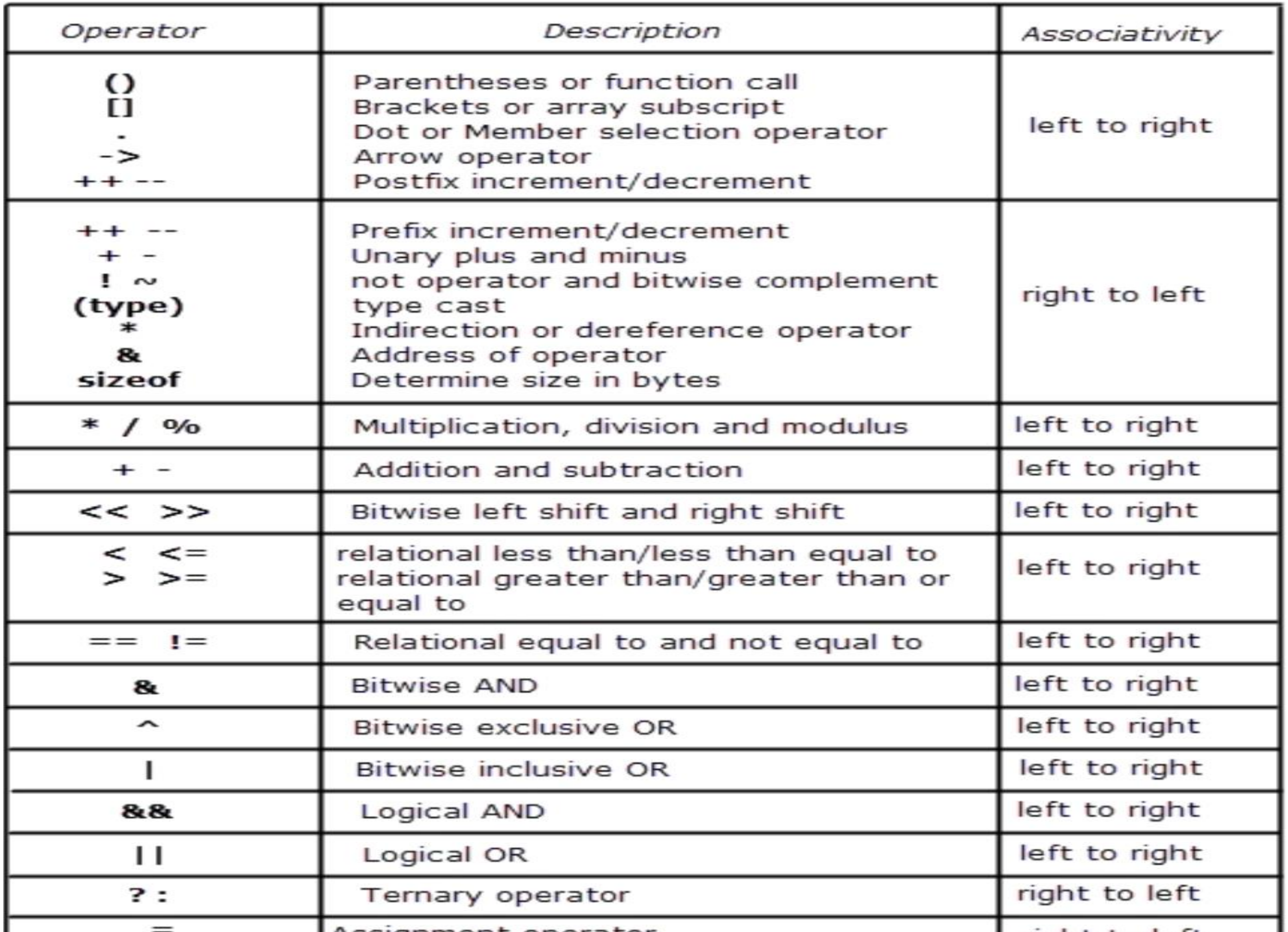
(Note: This program is based on the Operators Precedence and Associativity)

1. a+b / c+d
2. (a+b)*(a-b)
3. a+b/2
4. (a+b)/(c+d)
5. a+b*a-b
6. (a+b)/2

where a, b, c, d are integer variables.



Precedence order	Operator	Associativity
1	( ) [ ] →	Left to right
2	++ -- - (unary) ! ~ * & sizeof	Right to left
3	* / %	Left to right
4	+ -	Left to right
5	<< >>	Left to right
6	< <= > >=	Left to right
7	== !=	Left to right
8	& (bitwise AND)	Left to right
9	^ (bitwise XOR)	Left to right
10	| (bitwise OR)	Left to right



Operator	*Description*	*Associativity*
() **[]** **.** **->** **++ --**	Parentheses or function call Brackets or array subscript Dot or Member selection operator Arrow operator Postfix increment/decrement	left to right
++ -- **+ -** **! ~** **(type)** ***** **&** **sizeof**	Prefix increment/decrement Unary plus and minus not operator and bitwise complement type cast Indirection or dereference operator Address of operator Determine size in bytes	right to left
*** / %**	Multiplication, division and modulus	left to right
+ -	Addition and subtraction	left to right
<< >>	Bitwise left shift and right shift	left to right
< <= **> >=**	relational less than/less than equal to relational greater than/greater than or equal to	left to right
== !=	Relational equal to and not equal to	left to right
&	Bitwise AND	left to right
^	Bitwise exclusive OR	left to right
|	Bitwise inclusive OR	left to right
&&	Logical AND	left to right
||	Logical OR	left to right
? :	Ternary operator	right to left
=	Assignment operator	right to left

Practical-3.2

Write a program for each:

1. Implicit type casting
2. Explicit type casting

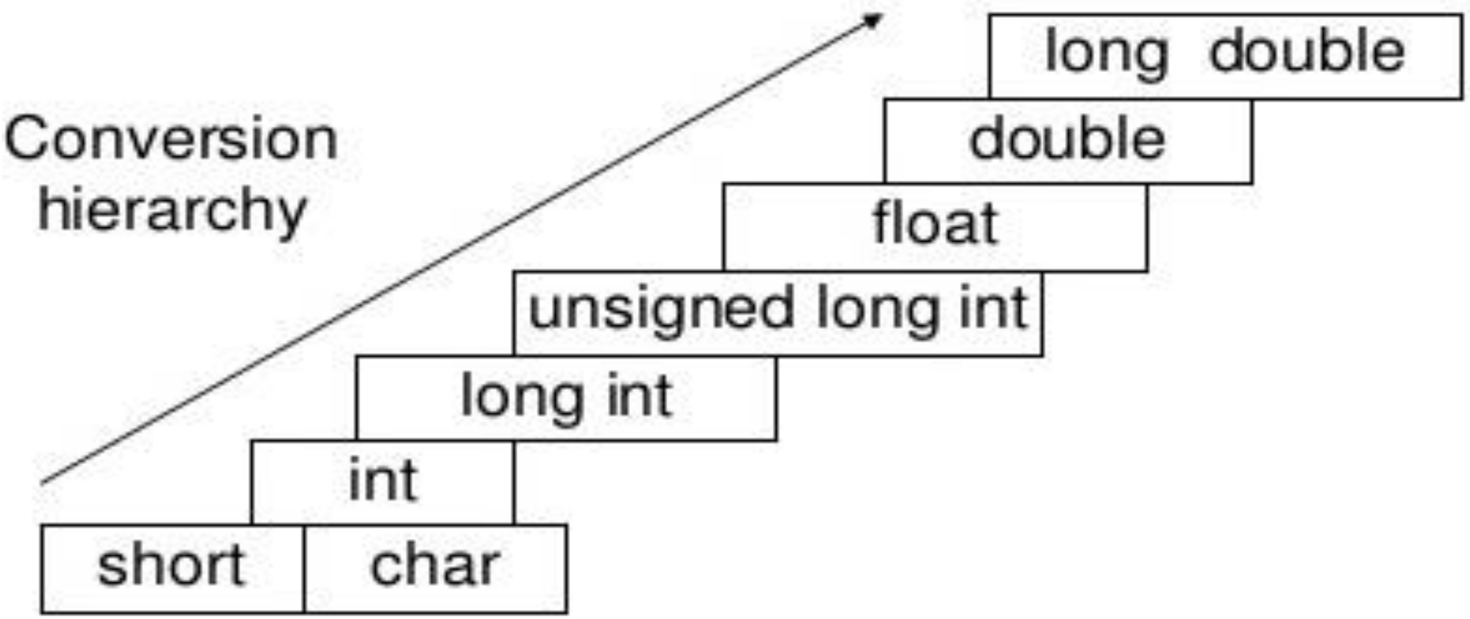
Type Casting in c

- Typecasting is converting one data type into another one.
- It is also called as data conversion or type conversion.
- two types of type casting operations:
  - Implicit type casting
  - Explicit type casting

Implicit type casting

- Implicit type casting means conversion of data types without losing its original meaning.
- This conversion is done by the compiler.
- When more than one data type of variables are used in an expression, the compiler converts data types to avoid loss of data.
- During conversion, strict rules for type conversion are applied.
- If the operands are of two different data types, then an operand having lower data type is automatically converted into a higher data type.

Implicit type casting



Conversion hierarchy

long double

double

float

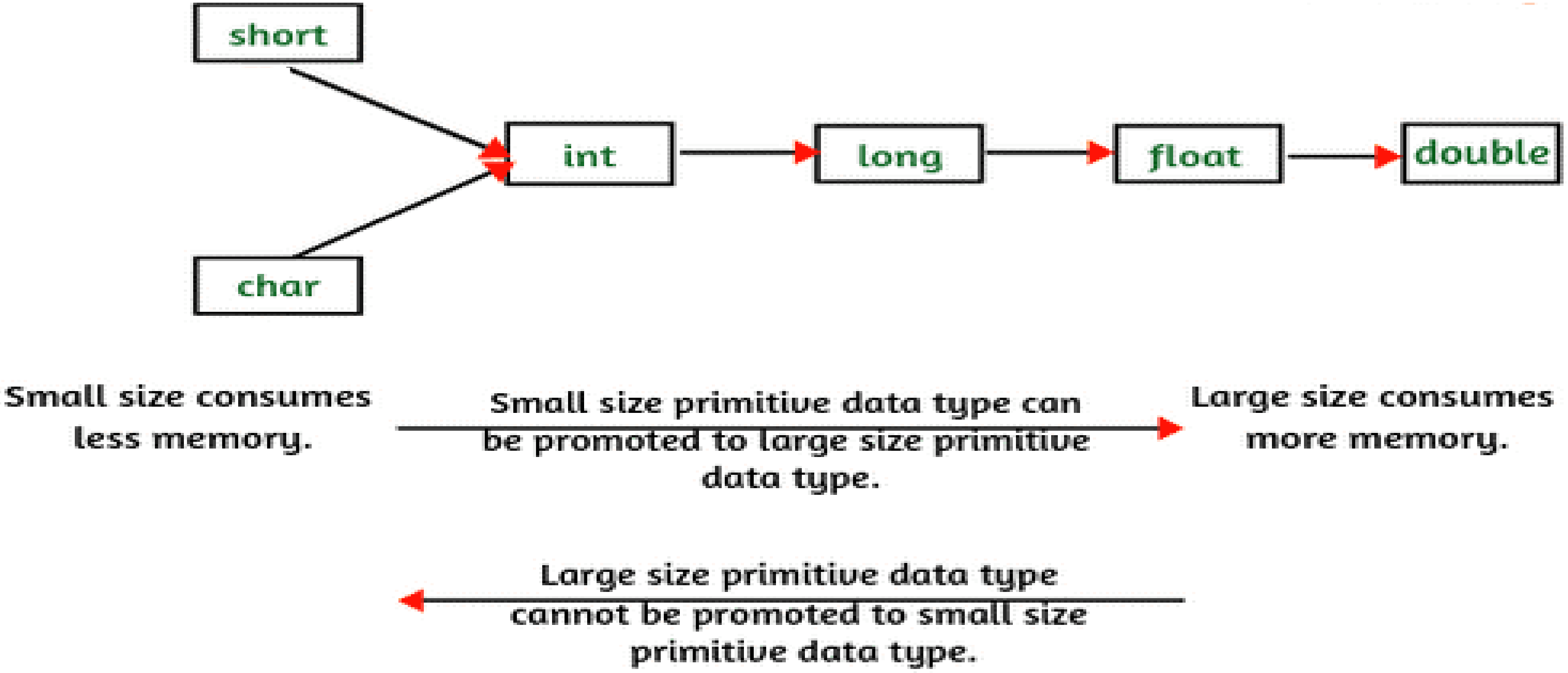
unsigned long int

long int

int

short char

Type casting



Small size consumes less memory.

Small size primitive data type can be promoted to large size primitive data type.

Large size consumes more memory.

Large size primitive data type cannot be promoted to small size primitive data type.

Example

```
#include <stdio.h>
Void  main()
{
    int number = 1;
    char character = 'k'; /*ASCII value is 107 */
    int sum;
    sum = number + character;
    printf("Value of sum : %d\n", sum );
 }
```

OUTPUT

```
Value of sum :108
```

Example

```c
#include <stdio.h>
 int main() {
 int a = 10;
 char b = 'S';
 float c = 2.88;
 a = a+b;
 printf("character to integer : %d\n",a);
 c = c+a;     //10.0+2.88
 printf(“Integer to float : %f\n",c);   //12.88
 return 0;
}
```

Explicit type Conversion

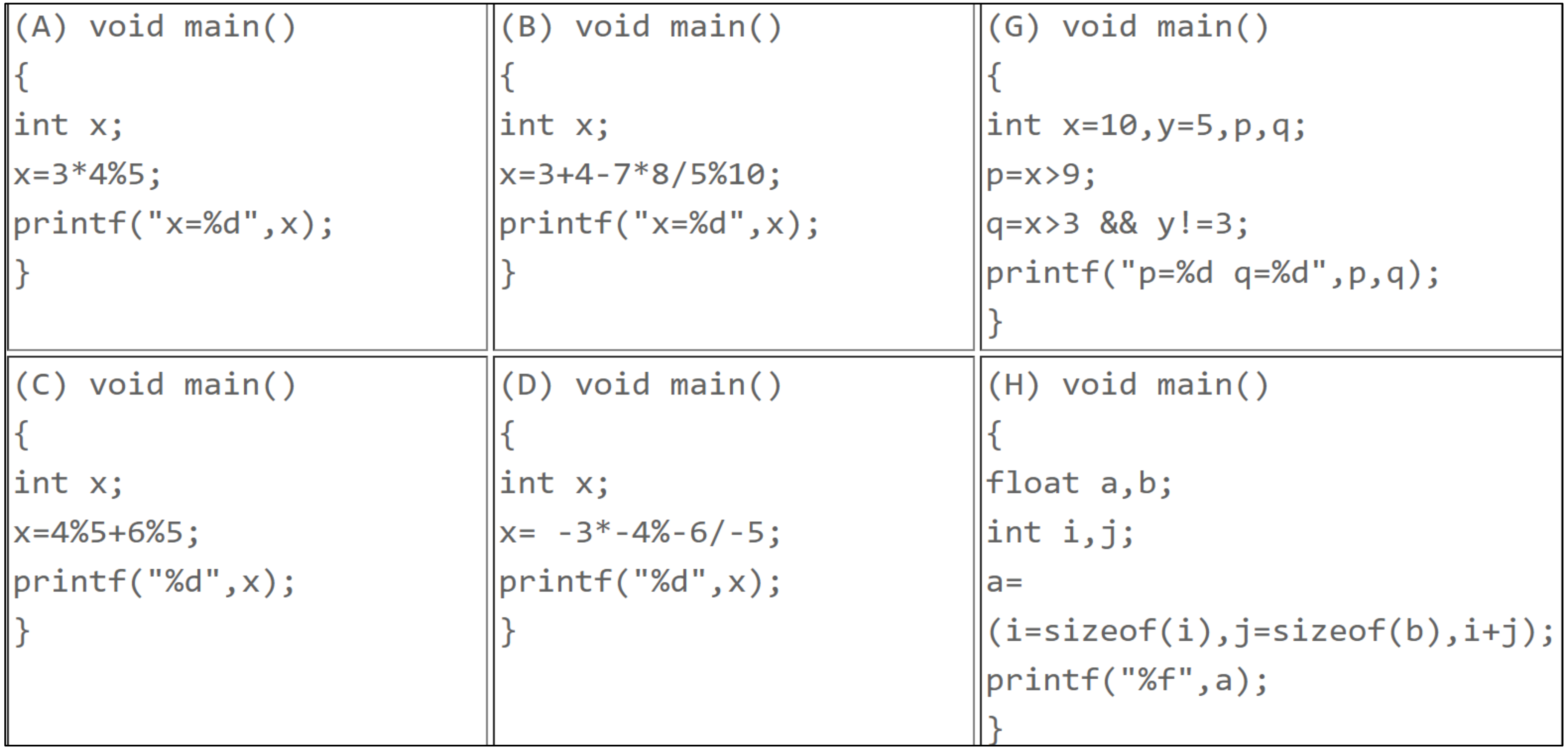
- This conversion is done by user.
- This is also known as typecasting.
- Data type is converted into another data type forcefully by the user.
- Syntax:
  - (Type) Expression/variable

Example

```
#include <stdio.h>
int main()
{
  float c = 5.55;
  int s = (int)c+1; //5+1
  printf("Explicit Conversion :%d\n",s);  //6
  return 0;
 }
```

Practical-3.3

- Check the out of following program and give your justification



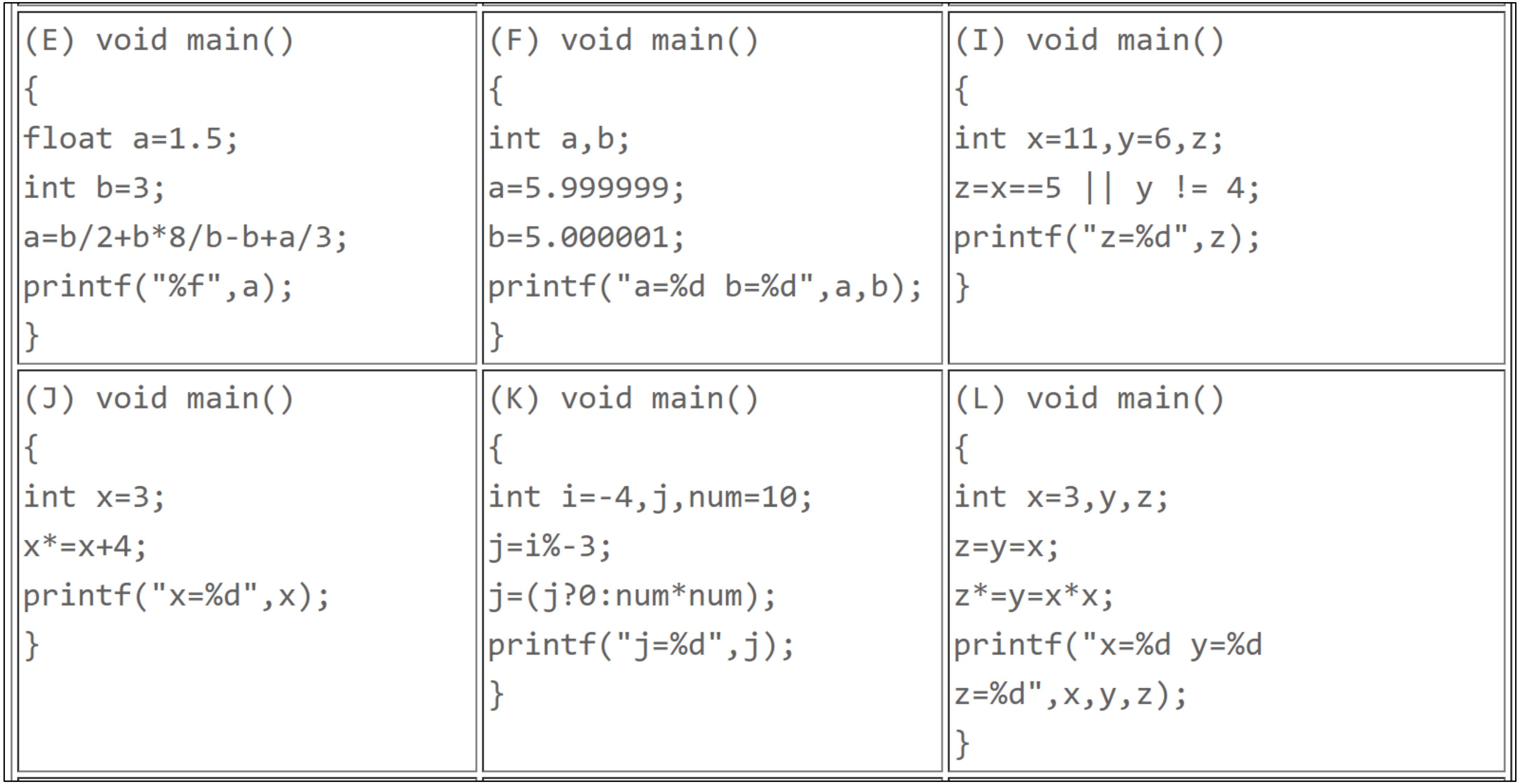
(A) void main()
{
int x;
x=3*4%5;
printf("x=%d",x);
}
	(B) void main()
	{
	int x;
	x=3+4-7*8/5%10;
	printf("x=%d",x);
	}
		(G) void main()
		{
		int x=10,y=5,p,q;
		p=x>9;
		q=x>3 && y!=3;
		printf("p=%d q=%d",p,q);
		}

(C) void main()
{
int x;
x=4%5+6%5;
printf("%d",x);
}
	(D) void main()
	{
	int x;
	x= -3*-4%-6/-5;
	printf("%d",x);
	}
		(H) void main()
		{
		float a,b;
		int i,j;
		a=
		(i=sizeof(i),j=sizeof(b),i+j);
		printf("%f",a);
		}

Practical-3.3 (Comma Operator)

- Used as a Separator –
- Used as an operator – The comma operator { , } is a binary operator that discards the first expression (after evaluation) and then use the value of the second expression.
- This operator has the least precedence.
- Example [for Exercise(L)]
- k=(3,4,7); // k is an integer
- printf("%d",k); //output : 7

Practical-3.3



```
(E) void main()
{
float a=1.5;
int b=3;
a=b/2+b*8/b-b+a/3;
printf("%f",a);
}
```

```
(F) void main()
{
int a,b;
a=5.999999;
b=5.000001;
printf("a=%d b=%d",a,b);
}
```

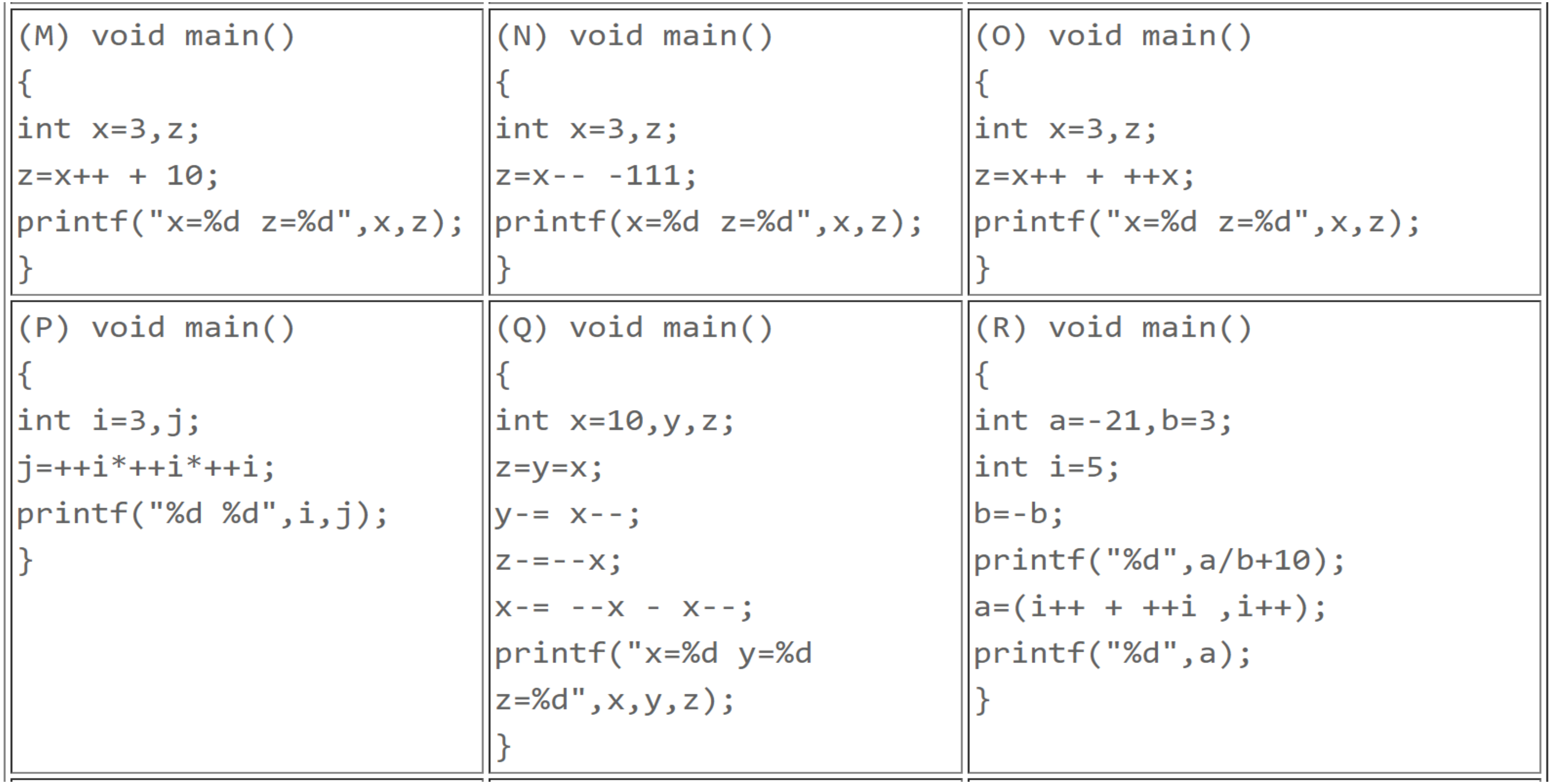
```
(I) void main()
{
int x=11,y=6,z;
z=x==5 || y != 4;
printf("z=%d",z);
}
```

```
(J) void main()
{
int x=3;
x*=x+4;
printf("x=%d",x);
}
```

```
(K) void main()
{
int i=-4,j,num=10;
j=i%-3;
j=(j?0:num*num);
printf("j=%d",j);
}
```

```
(L) void main()
{
int x=3,y,z;
z=y=x;
z*=y=x*x;
printf("x=%d y=%d
z=%d",x,y,z);
}
```

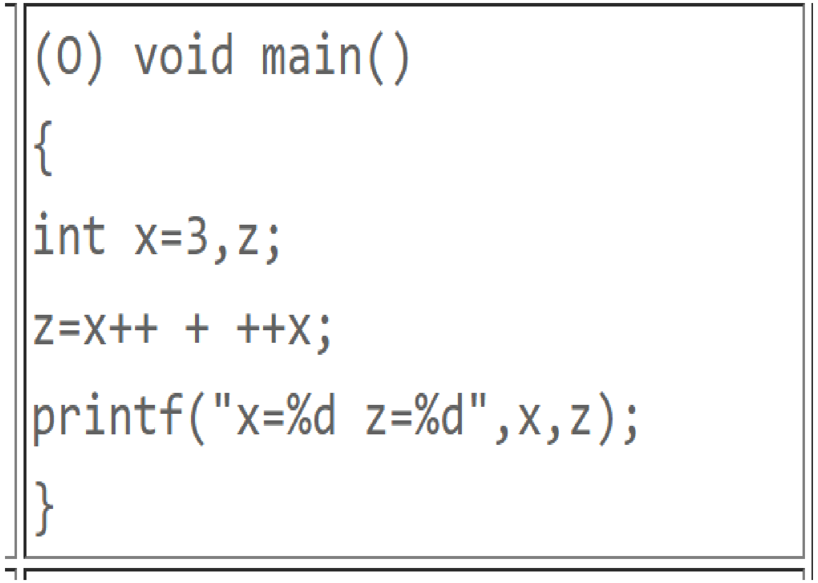
Practical-3.3



(M) void main()
{
int x=3,z;
z=x++ + 10;
printf("x=%d z=%d",x,z);
}
	(N) void main()
	{
	int x=3,z;
	z=x-- -111;
	printf(x=%d z=%d",x,z);
	}
		(O) void main()
		{
		int x=3,z;
		z=x++ + ++x;
		printf("x=%d z=%d",x,z);
		}

(P) void main()
{
int i=3,j;
j=++i*++i*++i;
printf("%d %d",i,j);
}
	(Q) void main()
	{
	int x=10,y,z;
	z=y=x;
	y-= x--;
	z-=--x;
	x-= --x - x--;
	printf("x=%d y=%d
	z=%d",x,y,z);
	}
		(R) void main()
		{
		int a=-21,b=3;
		int i=5;
		b=-b;
		printf("%d",a/b+10);
		a=(i++ + ++i ,i++);
		printf("%d",a);
		}

Practical-3.3

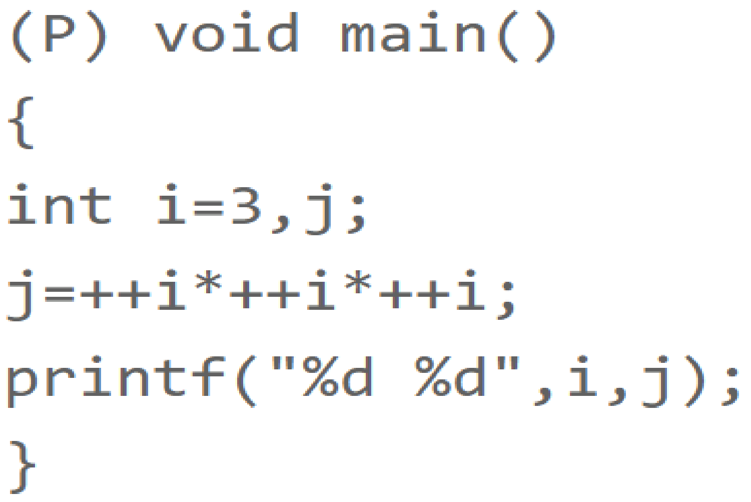


```
(O) void main()
{
int x=3,z;
z=x++ + ++x;
printf("x=%d z=%d",x,z);
}
```

- Z= x++ + ++x
- P=x++ //P=3 and x=4
- Q=++x //Q=5 and x=5
- Z=5+3

- x += (x++ + ++x) + x; //
- X+=(3+5)+5
- X=x+13

Practical-3.3



```
(P) void main()
{
int i=3,j;
j=++i*++i*++i;
printf("%d %d",i,j);
}
```

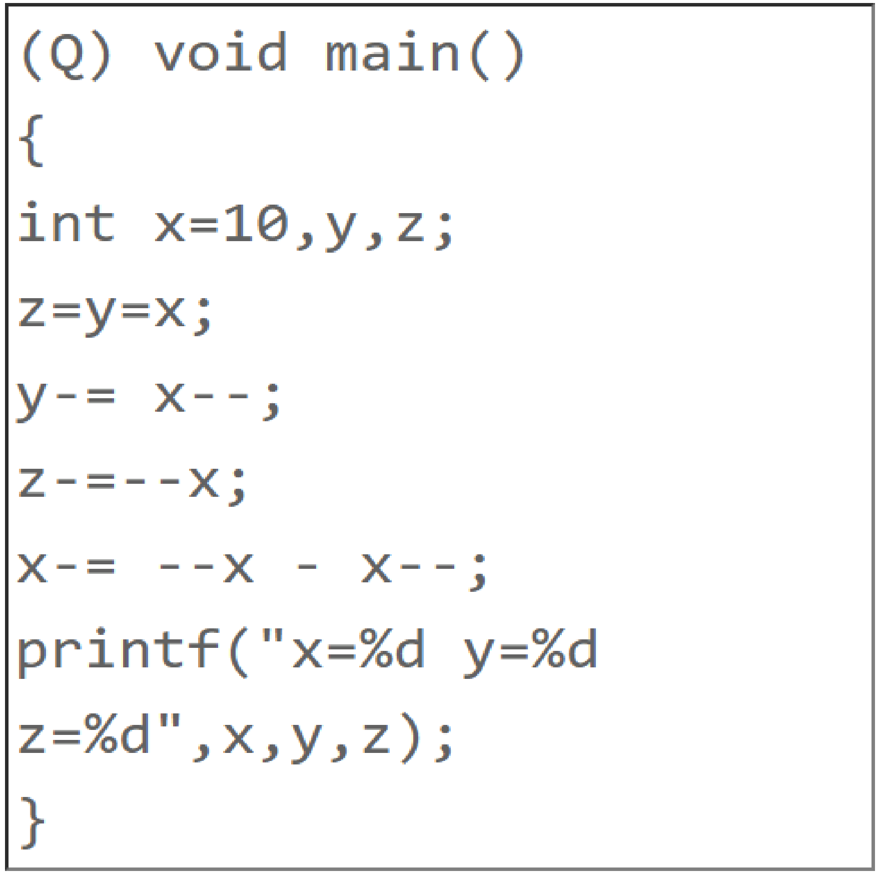
1. j=(++i * ++i) * ++i;
2. j = (++i)*(++i) // i=4 i=5 (5*5)
3. 25*(++i) //25*6=150

4. Example

   j= ++i + ++i + i++ ;//(5+5)+5

   j=++i + ++i + ++i ;//(5+5)+6

Practical-3.3



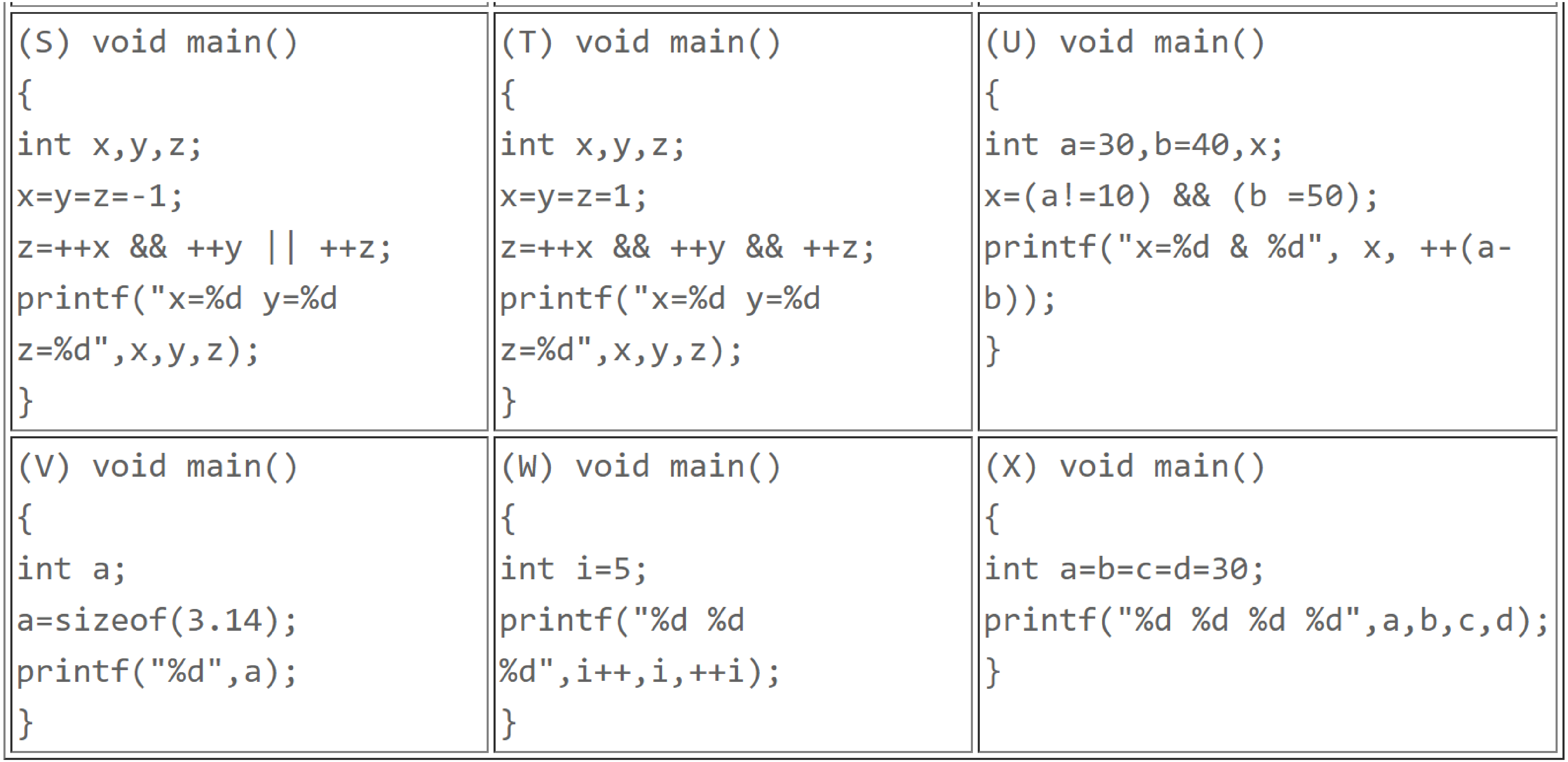
```
(Q) void main()
{
int x=10,y,z;
z=y=x;
y-= x--;
z-=--x;
x-= --x - x--;
printf("x=%d y=%d
z=%d",x,y,z);
}
```

1. z=y=x=10
2. x- - = 10 and y=y-10=0,x=9
3. --x=8 and z=z-8   z=10-8
4. x - = - -x – x- -

   1) --x x=7   4) (--x)=6

   2) x-- = 7   3) x=6
5. x=x-(-1)=6+1 =7

Practical-3.3



```
(S) void main()
{
int x,y,z;
x=y=z=-1;
z=++x && ++y || ++z;
printf("x=%d y=%d
z=%d",x,y,z);
}
```

```
(T) void main()
{
int x,y,z;
x=y=z=1;
z=++x && ++y && ++z;
printf("x=%d y=%d
z=%d",x,y,z);
}
```

```
(U) void main()
{
int a=30,b=40,x;
x=(a!=10) && (b =50);
printf("x=%d & %d", x, ++(a-
b));
}
```

```
(V) void main()
{
int a;
a=sizeof(3.14);
printf("%d",a);
}
```

```
(W) void main()
{
int i=5;
printf("%d %d
%d",i++,i,++i);
}
```

```
(X) void main()
{
int a=b=c=d=30;
printf("%d %d %d %d",a,b,c,d);
}
```

Practical-3.3

1. int x = 1,y = 0, z = 1;
2. Z = y && ++z || ++x;
3. //(false && not executed) || 2(True)
4. //Z=False|| True→ True
5. printf("x=%d,y=%d,z=%d",x,y,z);
6. //x=2 y=0 z=1

1. int x = 1,y = 1, z = 1;
2. z=++x || ++y && ++z;
3. //x=2 (true) || (not executed)
4. printf("x=%d,y=%d,z=%d",x,y,z);
5. //x=2 y=1 z=1

Practical-3.3

```
1.  int x = 1,y = 0, z = 1;
2.   z= ++y && ++z || ++x ;
3.   //( y=1 TRUE &&  executed z=2 TRUE) || not executed(OR operation)
4.  //z=True|| True→ True
5.  printf("x=%d,y=%d,z=%d",x,y,z);
6.  //x=1 y=1 z=1(in step-3 value is 2 but as per in step 3 ans is true so output is 1)
```

Practical-3.4

type the following program and justify its output:

```
void main ()
{
        int a=25,b=5,c,d,e,f,g;
        c = a + b;    d= a < b;    e= a % b;    f = a && b;    g = a << 2;
        printf(“a=%d \n b=%d \n c=%d\n ”,a , b, c);
        printf(“d=%d \n e=%d \n f=%d \n g=%d”, d, e, f, g);
 }
```

Practical-3.5

- Write a program for following:

1. To convert an angle to degree from radian.

   (use: degree = radian * 180 / PI)

2. To convert an angle to radian from degree.

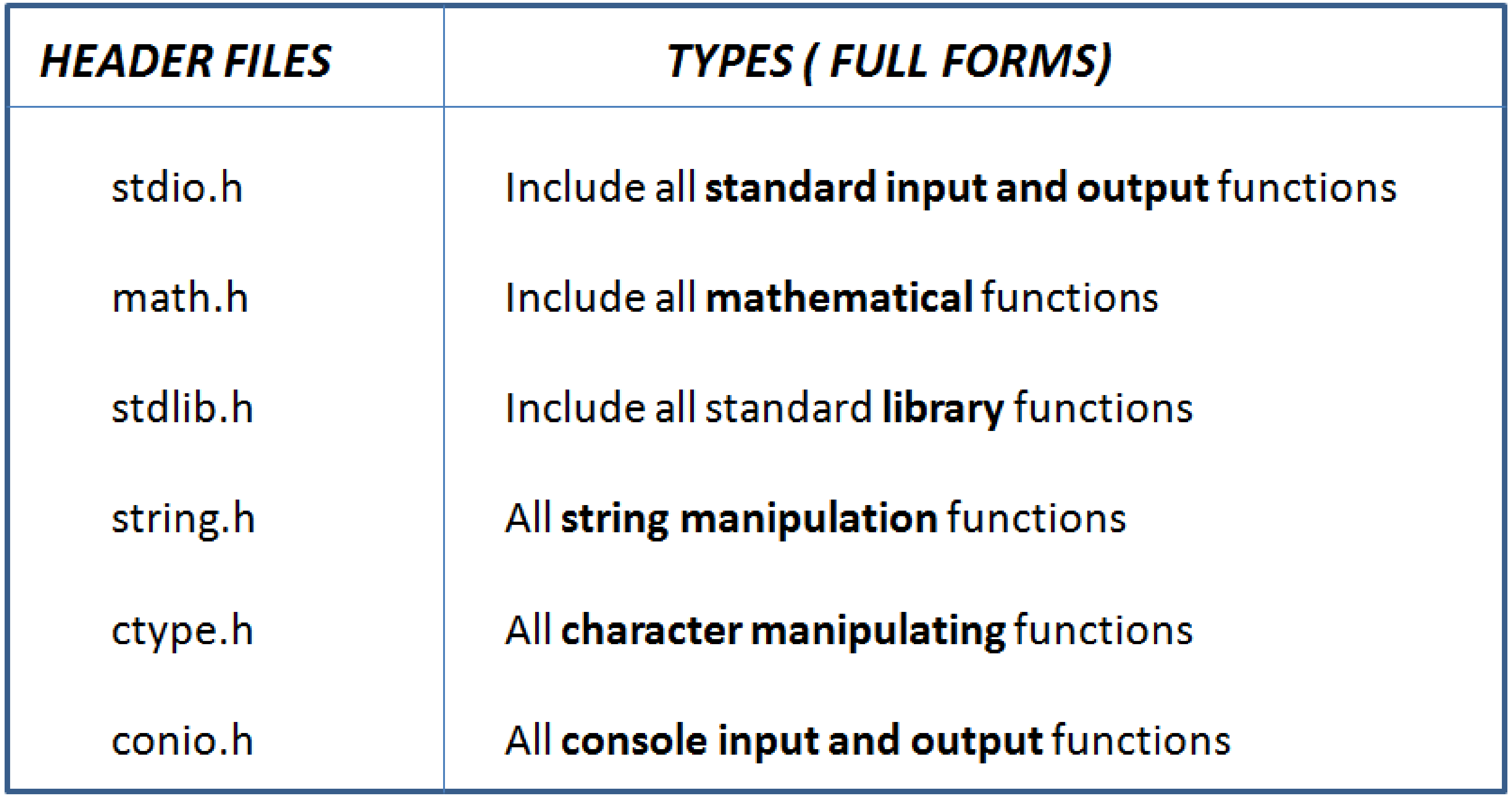
   Also find value of sin, cos and tan value of the entered value in the same program.

   (radian = degree * PI / 180)

Header files

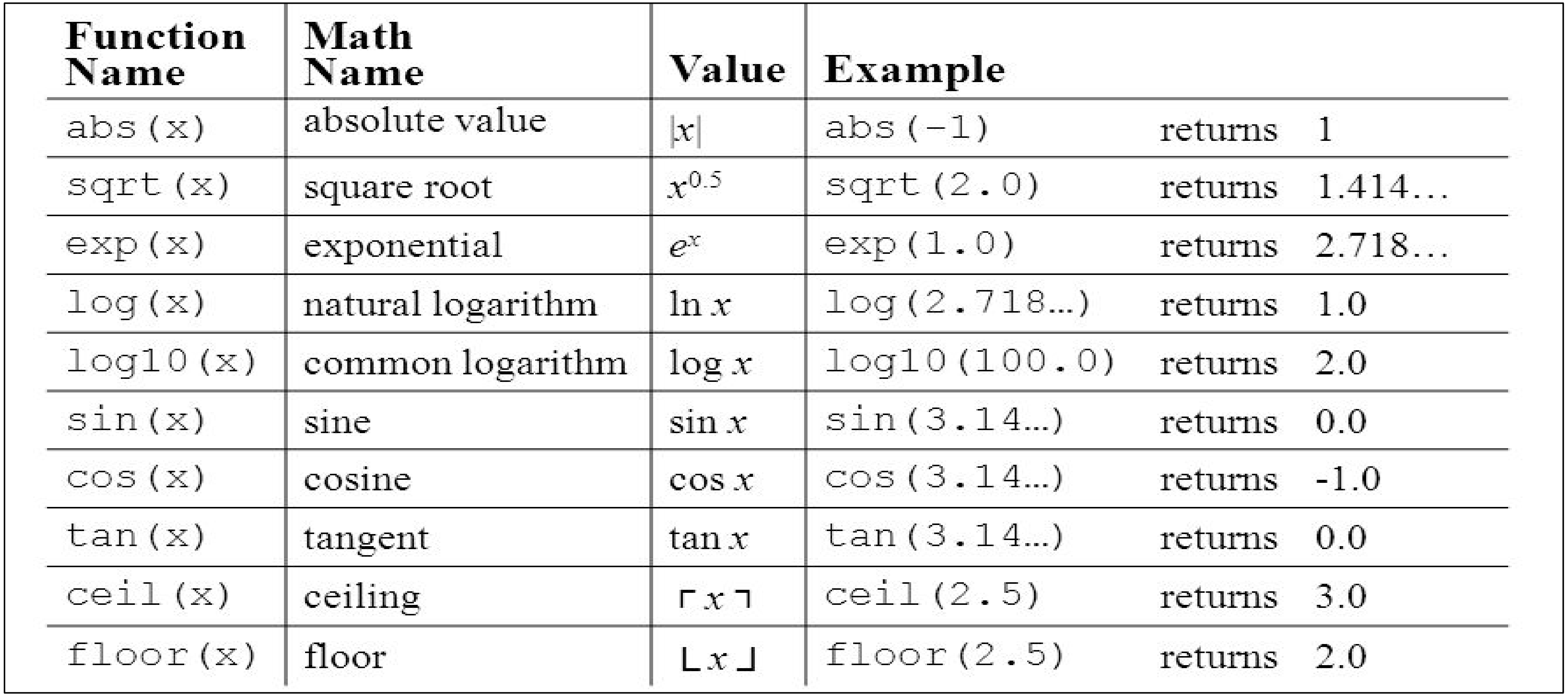
- A header file in C/C++ contains:
  - Function definitions
  - Data type definitions
  - Macros
- Header files offer these features by importing them into your program with the help of a preprocessor directive called **#include.**
- These preprocessor directives are responsible for instructing the C compiler that these files need to be processed before compilation.
- The default header file that comes with the C compiler is the stdio.h.

Header Files



HEADER FILES	***TYPES (FULL FORMS)***
stdio.h	Include all **standard input and output** functions
math.h	Include all **mathematical** functions
stdlib.h	Include all standard **library** functions
string.h	All **string manipulation** functions
ctype.h	All **character manipulating** functions
conio.h	All **console input and output** functions

<Math.h> functions



Function Name	Math Name	Value	Example		
`abs(x)`	absolute value	$\|x\|$	`abs(-1)`	returns	1
`sqrt(x)`	square root	$x^{0.5}$	`sqrt(2.0)`	returns	1.414…
`exp(x)`	exponential	e^x	`exp(1.0)`	returns	2.718…
`log(x)`	natural logarithm	$\ln x$	`log(2.718...)`	returns	1.0
`log10(x)`	common logarithm	$\log x$	`log10(100.0)`	returns	2.0
`sin(x)`	sine	$\sin x$	`sin(3.14...)`	returns	0.0
`cos(x)`	cosine	$\cos x$	`cos(3.14...)`	returns	-1.0
`tan(x)`	tangent	$\tan x$	`tan(3.14...)`	returns	0.0
`ceil(x)`	ceiling	$\lceil x \rceil$	`ceil(2.5)`	returns	3.0
`floor(x)`	floor	$\lfloor x \rfloor$	`floor(2.5)`	returns	2.0

Floor(x) function

- floor(x) : Returns the largest integer that is smaller than or equal to x

  (i.e : rounds downs the nearest integer).

- Examples of Floor:

  Input : 2.5 Output : 2

  Input : -2.1 Output : -3

Ceil(x) function

- **ceil(x) :** Returns the smallest integer that is greater than or equal to x

  (i.e : rounds up the nearest integer).

- Examples of Floor:

  Input : 2.5 Output : 3

  Input : -2.1 Output : -2

Convert angle to degree from radian

- #define PI 3.14
- Input : radian = Value
- degree = radian * (180/PI)
- Output : degree =Answer

Convert angle to radian from Degree

- #include <math.h>
- #define PI 3.14
- Input-: Degree = Value
- Radian = Degree * (PI / 180)
- Output-: radian is : Answer

  Sin(Degree) is : Answer

  Cos(Degree) is : Answer

  tan(Degree) is : Answer

Sample

- int i=3,a=3,j=3;
- int x =120;
- printf("%d %d %d",x, x++,++x);   ➔ 122   121   122
- printf("%d %d %d %d %d", i++, i--,++i, --i, i); ➔2 3 3 3 3
- printf("%d\t%d\t%d\n",i++,i,++i);   ➔ 4 5 5
- printf("%d %d %d\n",++a, a++,a); ➔5 3 5
- printf("%d\t %d\t %d\n", j,--j, j--);➔ 1 1 3