Operators and Expression

1 Explain operators available in C

An operator is a symbol that tells the compiler to perform certain mathematical or logical operation. C has rich set of operators as below,

1. Arithmetic Operators

Arithmetic operators are used for mathematical calculation. C supports following arithmetic operators

+	Addition or unary plus	a+ b (addition), +7 (unary plus)
-	Subtraction or unary minus	a – b (subtraction), -8 (unary minus)
*	Multiplication	a * b
/	Division	a / b
%	Modulo division	a % b (this operator can be used with only integer data type)

2. Relational Operators

Relational operators are used to compare two numbers and taking decisions based on their relation. Relational expressions are used in decision statements such as if, for, while, etc…

<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	is equal to
!=	is not equal to

3. Logical Operators

Logical operators are used to test more than one condition and make decisions

&&	logical AND (Both non zero then true, either is zero then false)
||	logical OR (Both zero then false, either is non zero then true)
!	logical NOT (non zero then false, zero then true)

4. Assignment Operators

Assignment operators are used to assign the result of an expression to a variable. C also supports shorthand assignment operators which simplify operation with

assignment.

=	Assigns value of right side to left side
+=	a += 1 is same as a = a + 1
-=	a -= 1 is same as a = a - 1
*=	a *= 1 is same as a = a * 1
/=	a /= 1 is same as a = a / 1
%=	a %= 1 is same as a = a % 1

5. **Increment and Decrement Operators**

These are special operators in C which are generally not found in other languages.

++	Increments value by 1. a++ is postfix, the expression is evaluated first and then the value is incremented. Ex. a=10; b=a++; after this statement, a= 11, b = 10. ++a is prefix, the value is incremented first and then the expression is evaluated. Ex. a=10; b=++a; after this statement, a= 11, b = 11.
--	Decrements value by 1. a-- is postfix, the expression is evaluated first and then the value is decremented. Ex. a=10; b=a--; after this statement, a= 9, b = 10. --a is prefix, the value is decremented first and then the expression is evaluated. Ex. a=10; b=--a; after this statement, a= 9, b = 9.

6. **Conditional Operator**

A ternary operator is known as Conditional Operator.

exp1?exp2:exp3 if exp1 is true then execute exp2

otherwise exp3 Ex: x = (a>b)?a:b; which is same as

```
if(a>b)
    x=a;
else
    x=b;
```

7. **Bitwise Operators**

Bitwise operators are used to perform operation bit by bit. Bitwise operators may not be applied to float or double.

&	bitwise AND
|	bitwise OR
^	bitwise exclusive OR
<<	shift left (1 bit shift left means multiply by 2)
>>	shift right (1 bit shift right means divide by 2)

8. Special Operators

&	Address operator, it is used to determine address of the variable.
*	Pointer operator, it is used to declare pointer variable and to get value from
,	Comma operator. It is used to link the related expressions together.
sizeof	It returns the number of bytes the operand occupies.
.	member selection perator, used in structure.
->	member selection perator, used in pointer to structure.

2 Explain conditional operator (ternary operator) with example.

- C provides special conditional operator (? :) to evaluate conditional expression in single line.
- It is also known as ternary operator because this is the only operator in C which requires three operands or expressions.
- Syntax: expr1? expr2 : expr3
- First of all expr1 is evaluated, if it is nonzero (true) then the expression expr2 is evaluated otherwise expr3 is evaluated. Only one of expr2 and expr3 is evaluated.
- Example: c = a>b? a : b; // If a is greater than b then a is assigned to c otherwise b.

```
#include<stdio.h>
#include<conio.h>
 void main()
{
          int a,b;
         a=10;
         b=15;
         (a>b)?printf("A is max"): printf("B is Max");
         getch();
}
```

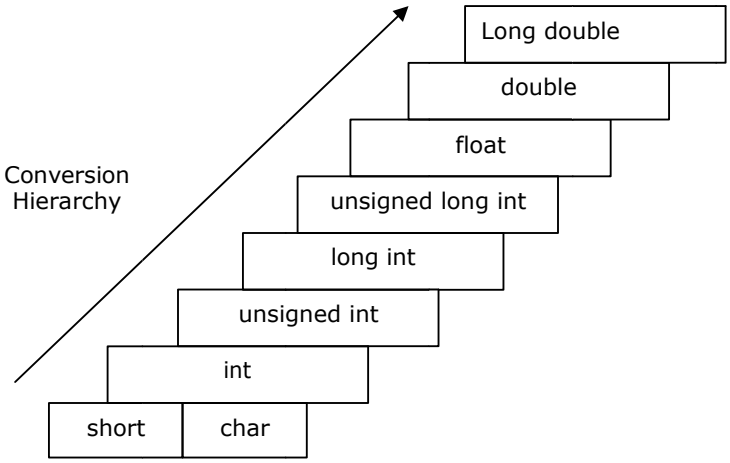
3 What is type conversion? Explain implicit and explicit type conversion with example.

a. When an operator has operands of different types, they are converted to a common type, this is known as type casting or type conversion.
b. Typecasting is making a variable of one data type to act like another data type such as an int to float.

Implicit Type Casting:

- C automatically converts any intermediate values to the proper type so that the expression can be evaluated without losing any significance.
- This automatic conversion is known as implicit type conversion.
- The lower type is automatically converted to the higher type before the operation proceeds. The result is higher type.
-

- Example:
  - If one operand is int and other is float then int will be converted to float.
  - If one operand is float and other is long double then float will be converted to long double.
- Thus conversion happens from low data type to high data type so that information is not lost. The conversion hierarchy is shown below.



Explicit Type Casting:

- Sometimes we want to force a type conversion in a way that is different from automatic conversion. The process of such a local conversion or casting is known as explicit casting.
- The general form of cast is or syntax is **(type-name) expression.**
- Where type-name is one of the standard C data type. The expression may be constant, variable, or an expression.

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int sum=47, n=10;
        float avg;
        avg=sum / n;
        printf("Result of Implicit Type Casting: %f", avg);
        avg=(float)sum / (float)n;
        printf("Result of Explicit Type Casting: %f", avg);
}
```

4 Explain operator precedence and associativity.

a. Precedence of an operator is its priority in an expression for evaluation.

b. Operator precedence is why the expression 5 + 3 * 2 is calculated as 5 + (3 * 2), giving 11, and not as (5 + 3) * 2, giving 16.

c. We say that the multiplication operator (*) has higher "precedence" or "priority" than the addition operator (+), so the multiplic tion must be performed first.

d. Associativity is the left-to-right or right-to-left order for grouping operands to operators that have the same precedence.

e. Operator associativity is why 8 - (3 - 2), giving 7.

f. We say that the subtraction operator (-) is "left associative", so the left subtraction must be performed first. When we can't decide by operator precedence alone in which order to calculate an expression, we must use associativity.

g. Following table provides a complete list of operator, their precedence level, and their rule of association. Rank 1 indicates highest precedance level and 15 is the lowest.

Precedence	Associativity	Operator	Description
1	Left to Right	()	Function Call
		[]	Array Element Reference
2	Right to Left	+, -	Unary Plus, Unary Minus
		++, --	Increment, Decrement
		!	Logical Negation
		~	Ones Complement
		*	Pointer Reference (Indirection)
		&	Address
		sizeof	Size of an object
3	Left to Right	*	Multiplication
		/	Division
		%	Modulus
4	Left to Right	+	Addition
		-	Subtraction
5	Left to Right	<<	Left Shift
		>>	Right Shift
6	Left to Right	<	Less than

		<= > >=	Less than or equal to Greater than Greater than or equal to
7	Left to Right	== !=	Equality Inequality
8	Left to Right	&	Bitwise AND
9	Left to Right	^	Bitwise XOR
10	Left to Right	|	Bitwise OR
11	Left to Right	&&	Logical AND
12	Left to Right	||	Logical OR
13	Right to Left	?:	Conditional Expression
14	Right to Left	==, *=, /=, %=, +=, -=, &=, ^=, |=, <<=, >>=	Assignment Operators
15	Left to Right	,	Comma Operator