CHAPTER-8

OPERATORS & EXPRESSIONS IN C++

TYPE A : VERY SHORT ANSWER QUESTIONS

1.	**What is the function of operators? What are arithmetic operators? Give their examples.**
Ans.	Operators represent specific operations. Arithmetic operators carry out arithmetic for C++. These are unary +, unary -, +, -, *, / and %. For example, 4 + 2, 19 % 6, -a etc.
2.	**How is 'unary +' operator different from '+' operator? How is 'unary -' operator different from '-' operator?**
Ans.	A 'unary +' gives the value of its operand whereas '+' operator gives sum of its operand's value. A 'unary –' changes the sign of its operand's value whereas '-' operator subtract the value of second operand from the value of first operand.
3.	**What are binary operators? Give examples of arithmetic binary operators.**
Ans.	Operators that act upon two operands are referred to as binary Operators. The +, -, *, /, % etc are binary arithmetic operators. For example, 4 + 20 results in 24, 3 * 4 evaluates to 12.
4.	**What does the modulus operator % do? What will be the result of 7.2%2.1 and 8%3?**
Ans.	The % operator finds the modulus of its first operand relative to the second. The result of 7.2%2.1 and 8%3 are 0.9 and 2 respectively.
5.	**What will be the result of a=5/3 if a is (i) float (ii) int?**
Ans.	(i) 1 (ii) 1

6.	**Assuming that res starts with the value 25, what will the following code fragment print out?**
	`cout<<res--;`
	`cout<<++res;`

Ans.	2525
7.	**What will be the value of j = --k + 2k (l=k, l++) if k is 20 initially?**
Ans.	95
8.	**What will be the value of P = P * ++J where J is 22 and P = 3 initially?**
Ans.	P = 69
9.	**What will be the value of following, if j = 5 initially?** **(i) (5* ++j)%6 (ii) (5* j++)%6**
Ans.	(i) 0 (ii) 1
10.	**A relational operator** **(a) assigns one operand to another (b) yields a boolean (logical) result** **(c) compares two operands (d) logically combines two operands**
Ans.	(c) compares two operands
11.	**Write an expression that uses a relational operator to return true if the variable total is greater than or equal to final.**
Ans.	`if (total >= final)` `  return true;`
12.	**Given that i = 4, j = 5, k = 4, what will be the result of following expressions?** **(i) i < k (ii) i < j (iii) i <= k (iv) i == j (v) i == k (vi) j > k (vii) j >= i (viii) j != i (ix) j != k (x) j <= k**
Ans.	(i) false (ii)true (iii) true (iv) false (v) true (vi) false (vii) true (viii) true (ix) true (x) false
13.	**What will be the order of evaluation for following expressions?** **(i) i + 5>= j - 6? (ii) s + 10 < p - 2 + 2q (iii) i < j <k >i >= n**
Ans.	(i) + , - , >= (ii) + , - , + , < (iii) < , < , < , >= Evaluation of expression depends on operator precedence and operator associativity rules.
14.	**What will be the result of the following if ans is 6 initially?** **(i) cout<<ans = 8; (ii) cout<<ans == 8;**
Ans.	(i) ans = 8 is 8 (ii) ans = = 8 is 0
15.	**What could be the possible error resulting in the wrong behavior of the following code?** `int r = -13, l = -12;`

	`cout<<"Result="` `<<r<fabs(l);`
Ans.	There is a use of '<' instead of '<<' which is invalid. Following is correct code: `cout<<"Result="<<r<<fabs(l);`
16.	**What is the function of logical operators? Write an expression involving a logical operator to test if marks are 55 and grade if 'B'.**
Ans.	Logical operators connect the relationships among values. Following expression involve logical operator: `if((marks==55)&&(garde=='B'))`
17.	**The && and || operators** **(a) compare two numeric values (b) combine two numeric values** **(c) compares two Boolean values (d) combine two Boolean values**
Ans.	(b) combine two numeric values
18.	**What is the order of evaluation in the following expressions:** **(i) a>b || b<d? (ii) x==y && y<=m? (iii) a>b && b<c || c<!d + 3?**
Ans.	(i) (a>b)|| (b<d) : 1st a>b , 2nd b<c , 3rd || (ii) (x==y) && (y<=m) : 1st a==b , 2nd b==c , 3rd && (iii) ((a>b) && (b<c)) || (c<((!d) + 3)) : Try to identify your self
19.	**What is the result of following expression: a>=b && (a+b)>a** **(i) a=3, b=0 (ii) a=7, b=7?**
Ans.	(i) false (ii) true
20.	**What is the result of following expression if (i) check=3, (ii) check=0, (iii) check=-3 ?** **(!check)**
Ans.	(i) false (ii) true (iii) false
21.	**Identify the order of evaluation in the following expression:** **4 * 5 + 7 * 2 - 8 % 3 + 4 && 4/2 - 1 + 4 || 2 - 4 || 2 - 4 + 6 * 2**
Ans.	(((4 * 5)+(7 * 2)-(8 % 3)+4) && (4 / 2) - (1 + 4)) || (2 - 4) || (2 - 4(6* * 2))
22.	**Write a statement that uses a conditional operator to set grant to 10 if speed is more than 68, and to 0 otherwise.**
Ans.	`grant = speed>68 ? 10 : 0`
23.	**What will be the result of following expression if (i) age = 25 (ii) age = 65 (iii) age = 85?** **age>65 ? 350 : 100**
Ans.	(i) 100 (ii) 100 (iii) 350
24.	**What will be the result of following expression if (i) ans = 700, val = 300 (ii) ans = 800, val = 700?** **ans – val <500 ? 150 : 50**
Ans.	(i) 150 (ii) 150
25.	**What will be the results of following expression if** **(i) ans = 700, val = 300 (ii) ans = 800, val = 700?** **ans - (val <500 ? 150 : 50)**
Ans.	(i) 550 (ii) 750
26.	**Write a statement to print the size of (i) double type on your machine. (ii) a long variable result.**

Ans.	`(i) cout<<sizeof(double);`
	`(ii) long result;`
	`cout<<sizeof result;`

27.	**What is the result of following expression?** **(i) y = (t = 4, t + 3);** **(ii) y = ((t = 4, t + 3), (t – 2, t * 3))**
Ans.	(i) y = 7 (ii) y = 15
28.	**What is an expression? How many types of expression does C++ support?**
Ans.	An expression is any valid combination of operators, constants, and variables. C++ supports three types of expressions: arithmetic, relational or logical, compound etc.
29.	**Given the following set of declarations:**

	`char ch;`
	`int i, j, k;`
	`float a, b, c;`
	`double p, q;`
	Which of the following are arithmetic expressions
	(i) ch = 40 (ii) i + j - k (iii) j % k (iv) a % b + c - i (v) p + q/b > i

Ans.	(i) Not an arithmetic expression (ii) An arithmetic expression (iii) An arithmetic expression (iv) Not an arithmetic expression (v) Not an arithmetic expression
30.	**Using the declaration of previous question identify the logical expression in the following expressions:** **(i) (a) (ii) (!a) (iii) a + b % c (iv) a && !a (v) a||!a**
Ans.	(ii), (iv) and (v) are logical expressions.
31.	**Write an equivalent C++ expressions for the following expressions:** **(i) ut +** $\frac{1}{2}$**ft²** **(ii)** $\sqrt{sin\ a + tan - 1\ a - e^{2x}}$ **(iii) |a| + b>=|b| + a** **(iv)** $\begin{pmatrix} 3x+5y & 8xy \\ 5x+3y & 2yx \end{pmatrix}$ **(v) e** $^{|2x2-4x|}$
Ans.	(i) ut + ((1/2)(f*t*t)) (ii) sqrt (sin(a) + atan(a) – exp(2*x)) (iii) ((fabs(a) + b) >= (fabs(b) + a) (iv)((((3*x) + (5*y))/((5*x) + (3*y))) – ((8*x*y)/(2*y*x))) (v) exp(fabs((2*x*x) – (4*x)))
32.	**What is type conversion? What is meant by implicit and explicit type conversion?**
Ans.	Type conversion: The process of converting one predefined type into another is called Type conversion. Implicit type conversion: An implicit type conversion is a conversion performed by the compiler without programmer's intervention. Explicit type conversion: An explicit type conversion is user-defined that forces an expression to be of specific type.
33.	**What is the process of type promotion? What is integral promotion?**
Ans.	In implicit conversion, all operands are converted up to the type of the largest operand, which is called type promotion. The conversion of integer types (i.e., char, short, enumerator, int) into int or unsigned int is called integral promotion.
34.	**What do you mean by type casting? What is type cast operator?**
Ans.	The explicit conversion of an operand to a specific type is called type casting and it is done usig type-cast operator () that is used as (type) expression where *type* is a valid C++ type to which the conversion is to be done.
35.	**What will be the resultant type of the following expression if ch represents a char variable, i is an int variable, fl is a float variable and db is a double variable?** `ch - i + db / fl - i * fl + db / i.`
Ans.	The resultant type of the above expression is double as double is the largest data type.
36.	**What will be the resultant type of the following expression if fl is a float variable and db is a double variable?** `(int) (fl + db)`
Ans.	The resultant type of the above expression is int.
37.	**Construct logical expression to represent the following conditions:** **(i) ch is a lowercase character (The ASCII range for lowercase is 97 – 122)** **(ii) a is odd but less than 57.**
Ans.	`(i) ch>=97 && ch<=122` `(ii) a%2!=0 && a<57`
38.	**Construct logical expression o represent the following conditions:** **(i) Either age is more than 70 or members are more than or equal to 8.** **(ii) grade is 'B' and exper is more than 3 years.**
Ans.	`(i) age>70 || members>=8`

	(ii) grade=='B' && exper>3
39.	**If a = 50 and b = 4 then determine the result of the following:** **(i) a += b (ii) a %= b (iii) a -= b (iv) a /= b** **(v) a *= b (vi) cout<<100/9; (vii) float pi = 22/7; cout<<pi;**
Ans.	(i) 54 (ii) 2 (iii) 46 (iv) 12.5 (v) 200 (vi) 11 (vii) 3
40.	**An arithmetic assignment operator combines the effect of what two operators?**
Ans.	An arithmetic assignment operator combines the effect of an arithmetic operator and an assignment operator.
41.	**Write a statement that uses an arithmetic assignment operator to subtract the value of variable *ans* by 17. Write the same statement without arithmetic assignment operator.**

Ans.	Using arithmetic assignment operator:
	`ans - = 17`
	Without arithmetic assignment operator:
	`ans = ans - 17`

42.	**Which header file must be included in the program to make use of C++'s built-in mathematical functions?**
Ans.	The Which header file *math.h* must be included in the program to make use of C++'s built-in mathematical functions

TYPE B : SHORT ANSWER QUESTIONS

1. What are arithmetic operators in C++? Distinguish between unary and binary arithmetic operators. Give examples for each of them.

Ans. The operators which are performed arithmetic operations are called arithmetic operators.

Unary arithmetic operators	Binary arithmetic operators
A unary operator requires a single operand.	A binary operators requires two operands.
unary+, unary-, ++, --, sizeof etc. are unary operators.	+, -, *, /, % etc. are binary operators.
For example, if a=2 then +a means 2	For example, 2 + 5 results in 7

2. What is the function of increment/ decrement operator? How many varieties do they come in? How are these two varieties different from one another?

Ans. The increment and decrement operators (++ and --) add 1 or subtract 1 from operand's value. They come in two forms: postfix and prefix.

Postfix version	Prefix version
It uses the value of the operand and then changes it.	It first changes its operand's value and then uses it.
For example, a++ or a--	For example, ++a or --a

3. Evaluate a + = a + ++a if a = 20 initially.

Ans.
```
Initially a = 20
        a += a + ++a
           = 20 + 21
           = 63
```

4. What role do the relational operators play? How is relational operator == different from =?

Ans. Relational operators compare the values of their operands. If the comparison is true, the relational expression results into the value 1 and to 0, if the comparison is false.

'==' Operator	'=' Operator
It is a testing operator.	It is a assignment operator.
For example, value == 3 It returns 1 if the comparison is true otherwise returns 0.	For example, value = 3 It assigns 3 to *value*.

5. Why should following operations be discouraged?
(i) comparisons between two floating point numbers.
(ii) comparisons between signed and unsigned numbers.

Ans. (i) Floating-point arithmetic is not as exact and accurate as the integer arithmetic is. For instance, 3 * 5 is exactly 15, but 3.25 * 5.25 is nearly equal to 17.06. The exact number resulting from 3.25 * 5.25 is 17.0625. Therefore, after any calculation involving floating-point numbers, there may be a small residue error. Because of this error, the equality and inequality comparisons on floating-point number should be avoided.
(ii) It is because if we compare a signed value with an unsigned value, the compiler will treat the signed value as

	unsigned. If the signed value is negative, it will be treated as an unsigned integer having the same bit pattern as the signed value, and the result of the comparison will be arbitrary.
6.	**How are relational operators and logical operators related to one another?**
Ans.	The relational operator compares the values of their operands and logical operators connect the relationship among values. For example, to check whether the age is more than 30 and experience is equal to 3 years following expression is used: (age>30) && (expr=3) Above code is a combination of relational and logical operators. Because logical AND operator (&&) has lower precedence than the relational operator, parentheses are not used in above expression.
7.	**Illustrate with an example that the unary negation operator ! is useful as a test for zero.**
Ans.	The unary negation operator ! is useful as a test for zero. For example, the condition if (check == 0) can be written as (!check). As unary negation operator negates the truth value of an expression, it will negates the value of the expression (!check) and test whether the variable *check* is equal to 0 or not.
8.	**Evaluate the following expressions:** **(i) x – y < z && y + z ? x || x – z <= y – x + z if x = 4, y = 7 and z = 10?** **(ii) (y) && (y – z) || !(2y + z – x) if x = 13, y = 14 and z = 5**
Ans.	(i) (((x – y) < z) && ((y + z) > x)) || ((x – z) <= (y – (x + z)) (((4 – 7) < 10) && ((7 + 10) > 4)) || (((4 – 10) <= (7 – (4 + 10))) ((1) && (1)) || (0) (1) || (0) =1 The result is 1 i.e., true (ii) **(y) && (y – z) || !(2y + z – x)** ((14) && (14 – 5)) || (!((2(14) + 5) – 13)) ((14) && (9)) || (!(28+5)-13) (1) || (0) = 0 The result is 0 i.e., false
9.	**State why are following expressions invalid?** **(i) asm = 5100 || val<35** **(ii) age > 70 && <90** **(iii) income >= 5000 ||&& val <500** **(iv) res! > 20 || x > 20.**
Ans.	(i) 'asm' is a keyword which cannot be used as a variable name. (ii) There should be a variable name after && operator. (iii) Both operator '||' and '&&' cannot be used together. (iv) Both operator '!' and '>' cannot be used together.
10.	**What are unary, binary and ternary operators? Classify the operators you have learnt so far into these three categories.**
Ans.	Unary operator: Operators that act on one operand are referred to as unary operators. Binary operator: Operators that act upon two operands are referred to as binary operators. Ternary operator: Operators that requires three operands are referred to as ternary operators.

Unary operator	Binary operator	Ternary operator
unary +, unary -, ++, --, sizeof	+, -, *, /, %	?: (the conditional operator)

11.	**Under what condition do the mathematical function log(), log10() and pow() produce an error?**
Ans.	The mathematical function log(), log10() produce a domain error if argument is negative and a range error occurs if the argument is zero. The function pow() produce a domain error if base = 0 and exp <= 0. Also if base < 0 and exp is not integer.
12.	**What are the purposes of following mathematical functions in C++?** **(i) atan() (ii) atan2() (iii) ceil() (iv) exp()**

Ans.

No.	Function	Prototype	Description
(i)	**atan()**	double atan(double arg)	The atan() function returns the arc tangent of *arg.*
(ii)	**atan2()**	double atan2(double b, double a)	The atan2() function returns the arc tangent of b/a.

(iii)	ceil()	double ceil(double num)	The ceil() function returns the smallest integer represented as a double not less than *num*.
(iv)	exp()	double exp(double arg)	The exp() function returns the natural logarithm e raised to the arg power.

13. Describe the working of these mathematical functions:
(i) fabs() (ii) floor() (iii) fmod()

Ans.

No.	Function	Prototype	Description	Example
(i)	fabs()	double fabs(double num)	The fabs() function returns the absolute value of *num*.	fabs(1.0) gives 1.0 fabs(-1.0) gives 1.0
(ii)	floor()	double floor(double num)	The floor() function returns the largest integer not greater than num.	floor(1.03) gives 1.0 floor(-1.03) gives -2.0
(iii)	fmod()	double fmod(double x, double y)	The fmod() function returns the remainder of the division *x/y*.	fmod(10.0,4.0) returns 2.0

14. What is meant by type conversion? How is implicit conversion different from explicit conversion?

Ans. The process of converting one predefined type into another is called Type conversion. An **implicit conversion is** a conversion performed by the compiler without programmer's intervention whereas, an explicit type conversion is user-defined that forces an expression to be of specific type.

15. Write arithmetic type conversion rules for float types and int types.

Ans.
1. If either operand if float then, other is converted to float.
2. Otherwise, the integral promotions are performed on both operands.
3. If one operand is a long int and the other unsigned int, then if a long int can represent all the values of an unsigned int, the unsigned int is converted to long int; otherwise both operands are converted to unsigned long int.
4. Otherwise, both operands are int.

16. Determine the data type of the expression

$$\begin{pmatrix} 100(1-pq) \\ (q+r) \end{pmatrix} - \begin{pmatrix} (p+r)/s \\ (long)(s+p) \end{pmatrix}$$

If p is an int, r is a float, q is a long and s is double.

Ans. Double type.

17. Determine the data type of the expression

$$\begin{pmatrix} 2x+3y \\ 5w+6z \end{pmatrix} - \frac{8p}{5q}^4$$

If x is it, y is long, w is float, z is double, p is short and q is long double.

Ans. Long double.

18. Evaluate the following C++ expressions where a, b, c are integers and d, f are floating point numbers. The value of a = 5, b = 3 and d = 15.
(a) f = a + b/a
(b) c = d * a + b
(c) c = (a++) * d + a
(d) f = (++b)* b – a
(e) c = a-(b++)*(--a)

Ans.

(a) f = a + b/a
```
f = a + b/a
  = 5 + 3/5
  = 5 + 0
  = 0
```

(b) c = d * a + b
```
c = d * a + b
  = 1.5 * 5 + 3
  = 7.5 + 3
  = 10.5
  = 10          (As c is integer)
```

(c) c = (a++) * d + a
```
c = (a++) * d + a
  = (5++) * 1.5 + 5
  = 5 *1.5 + 5 AS Post increment)
  = 7.5 + 5
  = 12.5
  = 12       (As c is integer)
```

(d) f = (++b)* b – a
```
f = (++b)* b – a
 = (++3)* 3 - 5
 = 4 * 3 - 5(AS Pre increment)
 = 12 - 5
 = 7
 = 7.0          (As f is float)
```

(e) c = a-(b++)*(--a)

```
c = a-(b++)*(--a)
  = 5- 3 * 4        (As b is post increment and a is pre decrement)
  = 5 - 12
  = -7
```

19. **Write the C++ equivalent expressions for the following:**
(a) volume = 3.1459 $r^2h/3$
(b) fn = 0.9 if x = 60

Ans.

```
a)volume = (3.1459*r*r*h)/3
b) if x = 60
          fn = 0.9
```

20. **Suppose A, B, C are integer variables A = 3, B = 3, C = -5 and X, Y, Z are floating point variables where X = 8.8, Y = 3.5, Z = -52. Determine the value of the following expressions:**
(a) A % C **(b) A * B/C**
(c) (A * C)%B **(d) X/Y**
(e) X/(X + Y) **(f) int(X) % int (Y)**

Ans. (a) 3 (b)-1 (c)0 (d)2.514286 (e)0.715447 (f)2

21. **What are the potential problems in type conversions?**

Ans.

No.	Conversion	Potential Problems
1.	Bigger floating-point type to smaller floating-point type (e.g., double to float)	Loss of precision. Original value may be out of range for target type, in which case result is undefined.
2.	Floating-point type to integer type.	Loss of fractional part. Original value may be out of range for target type, in which case result is undefined.
3.	Bigger integer type to smaller integer type (e.g., long to short)	Original value may be out of range for target type. Typically, just resulting in loss of information.

22. **Why do you think type compatibility is required in assigning values?**

Ans. When variable of one type are mixed with variables of another type, a type conversion will occur. In an assignment statement, the type conversion rule states: The value of the right side of the assignment is converted to the type of the left side. Therefore, the types of right side and left side of an assignment should be compatible so that type conversion can take place. The compatible data types are mathematical data types i.e., char, int, float, double. For example, the following statement

ch = x; (where ch is char and x is int)

converts the value of *x* i.e., integer to bit into *ch*, a character.

23. **Discuss the possible info-loss in common type conversions.**

Ans.

Target Type	Expression Type	Possible Info Loss
signed char	char	If value > 127, target is negative
char	short int	High-order 8 bits
char	int	High-order 8 bits
char	long int	High-order 8 bits
int	long int	High-order 16 bits
int	float	Fractional part and possibly more
float	double	Precision, result rounded
double	long double	Precision, result rounded

24. **Discuss the role and importance of C++ shorthand's. Do you see any benefit from them?**

Ans. C++ offers special shorthand's to simplify a certain type of assignment statement. The general form of a C++ shorthand is var oper = expression
where *var* is the target variable, *oper* is a binary operator and expression is the *expression* that is to be operated to the previous value of var. for example, x -= 2 means x = x - 2
C++ shorthand uses arithmetic assignment operators. One important and useful thing about such arithmetic assignment operator is that they combine an arithmetic operator and assignment operator, and eliminate the repeated operand thereby facilitate a condensed approach.

25. **Discuss the function of conditional operator, sizeof operator and comma operator with examples.**

Ans.	**The conditional operator:** It requires three operands, that is, it is a ternary operator. Its general form is expr1? expr2: expr3. It gives the value of expr2 if expr1 evaluates to true otherwise it gives the value of the expr3. For example, 13<20? 1: 2 evaluates to 1. **The sizeof operator:** Returns the size of a variable or a data-type. It can be used in two forms: sizeof *var* (where var is a declared variable) and sizeof(type) (where *type* is a C++ data type. For example, sizeof(int) returns 2. **The comma operator:** Strings together several expressions which are evaluated from left-to-right and the value of the rightmost expression becomes the value of the total comma-separated expression. For example, y = (x=2, x-1); first assign *x* the value 2 and the assigns *y* the value x-1 i.e., 1.
26.	**An expression uses arithmetic, relational and logical operators together. How would you determine the type (arithmetic, logical) of the expression?**
Ans.	Type of operators used in an expression determines the expression type. If an expression uses arithmetic, relational and logical operators together, we have to divide the expression in sub-expression and then decide the type of expression for sub-expression.
27.	**Discuss the benefits and loopholes of type casting.**
Ans.	By type casting we can convert variable from one data type to another data type, which give us advantage to manipulate variable as per our convenience. There few loopholes in type casting which are as follow – 1. Assigning a value of larger data type to a smaller data type may result in losing some precision. 2. Conversion of bigger floating-point type to smaller floating-point type may result in loss of precision. 3. Conversion of floating-point type to integer type may result in loss of fractional part. 4. Conversion of bigger integer type to smaller integer type may result in loss of information. Original value may be out of range for target type.
28.	**Write a program which will raise any number X to a positive power n. obtain values of x and n from user.**

Ans.	`#include<iostream.h>`
	`#include<conio.h>`
	`#include<math.h>`
	`void main()`
	`{ clrscr();`
	`int X,p,res;`
	`cout<<"Enter X: ";`
	`cin>>X;`
	`cout<<"Enter p: ";`
	`cin>>p;`
	`res=pow(X,p);`
	`cout<<"\t"<<res;`
	`getch();`
	`}`

29.	**Write a C++ program to input principle amount and time. If time is more than 10 years, calculate the simple interest with rate 8%. Otherwise calculate it with rate 12% per annum.**

Ans.	`#include<iostream.h>`
	`#include<conio.h>`
	`void main()`
	`{ clrscr();`
	`float i,p,r,n;`
	`cout<<"Enter Price: ";`
	`cin>>p;`
	`cout<<"Enter Duration: ";`
	`cin>>n;`
	`if(n>10)`
	`i=(p*8*n)/100;`
	`else`
	`i=(p*12*n)/100;`
	`cout<<"\t"<<"Simple Interest: "<<i;`
	`getch();`
	`}`

30. **Write a C++ program to input a number. If the number is even, print its square otherwise print its cube.**

Ans.

```
#include<iostream.h>
#include<conio.h>
void main()
{     clrscr();
      int n,res;
      cout<<"Enter n: ";
      cin>>n;
      if(n%2==0)
            res=n*n;
      else
            res=n*n*n;
      cout<<"\t"<<res;
      getch();
}
```

31. **Write a C++ program to input a number. If the number n is odd and positive, print its square root otherwise print n^5.**

Ans.

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
void main()
{     clrscr();
      int n,res;
      cout<<"Enter n: ";
      cin>>n;
      if((n%2!=0) && (n>0))
            res=sqrt(n);
      else
            res=n*n*n*n*n;
      cout<<"\t"<<res;
      getch();
}
```

32. **Write a C++ program to input choice (1 or 2). If choice is 1, print the area of circle otherwise print the perimeter of circle. Accept the radius of circle from user.**

Ans.

```
#include<iostream.h>
#include<conio.h>
void main()
{     clrscr();
      int ch;
      float r,area,peri;
      cout<<"Enter your choice: ";
      cin>>ch;
      switch(ch)
      {
            case 1: cout<<"Enter radius: ";
                  cin>>r;
                  area=3.14*r*r;
                  cout<<"Area of circle: "<<area;
                  break;
            case 2:  cout<<"Enter radius: ";
                  cin>>r;
                  peri=2*3.14*r;
                  cout<<"perimeter of circle: "<<peri;
                  break;
            default: cout<<"Wrong choice";
      }
```

```
      getch();
```

33. **Write a C++ program to input three integers and print the largest of three.**

Ans.

```
#include<iostream.h>
#include<conio.h>
void main()
{     clrscr();
      int a,b,c;
      cout<<"Enter a, b, and c: ";
      cin>>a>>b>>c;
      if((a>b) && (a>c))
            cout<<"Largest: "<<a;
      else if((b>a) && (b>c))
            cout<<"Largest: "<<b;
      else
            cout<<"Largest: "<<c;
      getch();
}
```

34. **Write a C++ program to input a student type ('A' or 'B'). if the student type is 'A' initialize the college account with Rs. 200/- otherwise initialize the hostel account with Rs. 200/-.**

Ans.

```
#include<iostream.h>
#include<conio.h>
void main()
{     clrscr();
      int clg_acc,hos_acc;
      char type;
      cout<<"Enter Student Type ('A' or 'B'):  ";
      cin>>type;
      if(type=='A')
      {     clg_acc=200;
            cout<<"College account is initialized with Rs."<<clg_acc;
      }
      else
      {     hos_acc=200;
            cout<<"Hostel account is initialized with Rs."<<hos_acc;
      }
      getch();
}
```

35. **Write a program that reads in a character <char> from the keyboard and then displays one of the following messages:**
(i) If <char> is a lower case letter, the message
(The upper case character corresponding to <char> is ….")
(ii) If <char> is a upper case letter, the message
(The lower case character corresponding to <char> is ….")
(iii) If <char> is not a letter, the message <chr> is not a letter",
Hint. You will need to refer to a table of ASCII characters. Lowercase letters have ASCII value range 97-122. uppercase letters have ASCII value range 65-90.

Ans.

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
void main()
{     clrscr();
      char ch;

      cout<<"Enter character:  ";
      cin>>ch;
      int code=ch;
```

```
      if(code>=97 && code<=122)
      {
            cout<<"The upper case character corresponding to "<<ch<<" is:
"<<(char)(code-32);
      }
      if(code>=65 && code<=90)
      {
            cout<<"The lower case character corresponding to "<<ch<<" is:
"<<(char)(code+32);
      }
      if((code<65 || code>90) && (code<97 || code>122))
      {
            cout<<"The "<<ch<<" is not a letter";
      }
      getch();
}
```

TYPE C : LONG ANSWER QUESTIONS

1.	**Discuss all operators and their types (with examples) that you have learnt so far.**
Ans.	Operators are the symbols that represent specific operations. 1. Arithmetic Operators: Carry out arithmetic operations. (a) Unary Operators: Operators that act on one operand. (i) Unary +: Gives the value of its operand. For example, if a = 5 then +a means 5. (ii) Unary -: Changes the sign of its operand's value. For example, if a = 5 then +a means -5. (b) Binary Operators: Operators that act upon two operands. (i) Addition Operator (+): Gives sum of its operand's value. For example, 4 + 20 results in 24. (ii) Subtraction Operator (-): Subtracts the value of second operand from the value of its first operand. For example, 14 – 3 evaluated to 11. (iii) Multiplication Operator (*): Gives the product of its operands' value. For example, 3 * 4 evaluates to 12. (iv) Division Operator (/): Divides the first operand by second. For example, 100/5 evaluates to 20. (v) Modulus Operator (%): Gives the remainder after dividing first operand by second. For example, 19 % 6 evaluates to 1. 2. Increment/Decrement Operators (++, --): Add 1 or subtract 1 from operand's value. The increment and decrement operators come in two forms: (i) Postfix version: First uses the value of the operand and then changes it. For example, a++, b— (ii) Prefix version: First changes its operand's value and then uses it. For example, ++a, --b 3. Relational Operators: Compare the values of their operands. These are <, >, <=, >=, == and != i.e., less than, greater than, less than or equal to, greater than or equal to, equal to and not equal to respectively. It returns boolean truth or false value. For example, 6==5 returns false, 6>5 returns true. 4. Logical Operators: Connect the relationship among values. (i) Logical OR (||): Evaluates to true if either of its operands evaluate to true. For example, (6<=6) || (5<3) returns true. (ii) Logical AND (&&): Evaluates to true if both of its operands evaluate to true. For example, (6<=6) || (5<3) returns false. (iii) Logical NOT(!): Negates the truth value of its operand. For example, !(6<=6) returns false. 5. Conditional Operator (?:): The general form of conditional operator is expr1? expr2: expr3. It gives the value of expr2 if expr1 evaluates to true otherwise it gives the value of the expr3. For example, 6>4? 9: 7 evaluates to 9. 6. Some other Operators: (a) The sizeof operator: Returns the size of a variable or a data-type. For example, sizeof(int) returns 2. (b) The Comma operator: Strings together several expressions which are evaluated from left-to-right and the value of the rightmost expression becomes the value of the total comma-separated expression. For example, b = (a=3, a+1); first assign a the value 3 and the assigns b the value a+1 i.e., 4.
2.	**What types of expressions can be formed in C++? Discuss each one of them with examples.**

Ans. Following types of expression can be formed in C++:

1. Arithmetic Expression: It can either be integer expression or real expression or mixed-mode expression.

(a) Integer expression: Combination of arithmetic operators, integer constants and/or integer variables. For example,

```
const count = 30;
int I, J, K, X, Y, Z;
```

(i) I (ii) –J (iii) K + X – Y + count

(b) Real expression: Combination of arithmetic operators, real constants and/or real variables. For example,

```
const bal = 250.53;
float qty, amount, value;
double fin, inter;
```

(i) qty/amount (ii)qty * value (iii) (amount + qty * value) - bal

(c) Mixed-mode expression: Combination of arithmetic operators, integer and/or real constants/variables.

2. Logical Expression: The expressions that results into 0 (false) or 1 (true) are called logical expressions. Logical expressions make use of logical and relational operators. For example, x>y, (y+z)>=(x/z)

3. What is the function of type conversion? Write the standard conversion rules of C++.

Ans. The type conversion converts one predefined type into another. Following are the standard conversion rules of C++:

1. If either operand is of type long double, the other is converted to long double.
2. Otherwise, if either operand is of type double, the other is converted to double.
3. Otherwise, if either operand is float, the other is converted to float.
4. Otherwise, the integral promotions are performed on both operands. The process of integral promotion is described below:
   A char, a short int, enumerator or an int may be used as an integer type. If an int can represent all the values of the original type, the value is converted to int; otherwise it is converted to unsigned int. this process is called integral promotion.
5. Then, if either operand is unsigned long., the other is converted to unsigned long.
6. Otherwise, if one operand is a long int and the other unsigned int, then if a long int can represent all the values of an unsigned int, the unsigned int is converted to long int; otherwise both operands are converted to unsigned long int.
7. Otherwise, if either operand is long, the other is converted to long.
8. Otherwise, if either operand is unsigned, the other is converted to unsigned.
9. Otherwise, both operands are int.

4. Discuss the role of assignment operator = in variable initialization and C++ shorthands. Support your answer with examples.

Ans. An assignment statement assigns value to a variable. The general form of an assignment is as below:

```
a = cve;
```

where *a* is the target variable and *cve* either be a constant or variable or an expression.

The symbol "=" is called the assignment operator. The assignment operator = returns the value of the assignment as well as actually assigning the value to the left hand operand. Because of that, assignment can be chained together. This can be useful when assigning the same value to a number of items, for example,

```
X= Y= Z= 13;
```

This statement assigns the value 13 to X, Y, and Z.

C++ offers special shorthands to simplify a certain type of assignment statement. The general form of a C++ shorthand is

```
var oper = expression
```

where *var* is the target variable, *oper* is a binary operator and expression is the *expression* that is to be operated to the previous value of var. for example,

```
c* = 10 means c = c * 10
```

and d/ = 2 means d = d / 2