1.

```
#include<stdio.h>

void main(){
        int i = -5;
        int num = i % 4;
        printf("%d\n", num);
}
/*
   A. Compile time error
   B. -1
   C. 1
   D. None

*/
```

/* Ans = B. -1
If the divident is negative then the reminder is also a negative num
e.g 1.-10 % 6 = -4
2 . -5 % -4 = -1
% a operator always have negtive value if divident is negative or if both are negtive.
***/**

2.

```
#include<stdio.h>
void main(){
        int i = 5;
        int num1 = i / -4;
        int num2 = i % -4;
        printf("%d %d\n", num1, num2);

}
/*
   A. Compile time error
   B. -1  1
   C. 1  -1
   D. Run time error
*/
```

/* Ans = C
/(Divide) operator:
1. if divisor < 0 and divident > 0 then ans is negtive
e.g 5/-4 = -1
2. if divident < 0 and divisor > 0 then ans is negative
e.g. -5 /4 = -1
3. if divident < 0 and divident < 0 then ans is negative
e.g -5 / -4 = 1
4. if divindet > 0 and divident > 0 then ans is positive
e.g 5 /4 =1

%(Mod) operator:

1. if divisor < 0 and divident > 0 then ans is negtive
e.g 5%-4 = 1
2. if divident < 0 and divisor > 0 then ans is negative
e.g. -5 %4 = -1
3. if divident < 0 and divident < 0 then ans is negative
e.g -5 % -4 = 1
4. if divindet > 0 and divident > 0 then ans is positive
e.g 5 % 4 =1
***/**

3.

```
#include<stdio.h>
void main(){
	int num = 7;
	num = num / 4;
	printf("%d\n", num);
}
```

```
/*
   A. Run time error
   B. 1
   C. 3
   D. Compile time error
*/
```

/* Ans : B
/ operator gives Quotient
num = num /4
= 7 / 4
= 1
***/**

4.

```
#include<stdio.h>
void main(){
	int num = 4 * 5 / 2 + 9;

	printf("%d\n",num);
}
```

```
/*
   A. 6.75
   B. 1.85
   C. 19
   D. 3
*/
```

/* Ans = C
num = 4 * 5 / 2 + 9 // there operator are used here. * , / and + . * and / have highest priority than +
= 20 / 2 + 9 // first * operation evaluated
= 10 + 9 // then /
= 19 // then +
***/**

5.

```
#include<stdio.h>
void main(){
        int x = 4.3 % 2;
        printf("Value of x is %d", x);
}
/*
   A. Value of x is 1.3
   B. Value of x is 2
   C. Value of x is 0.3
   D. Compile time error
*/
```

/*

Ans :D

Error: invalid operands to binary %(have double and int)

We can not apply mod operator on float and double values

*/

6.

```
#include<stdio.h>
void main(){

        int x = 7 % 4 * 3 / 2;
        printf("Value of x is %d", x);
}
/*
   A. Value of x is 1
   B. Value of x is 2
   C. Value of x is 4
   D. Compile time error
*/
```

/*

Ans : Value of X is 4

x = 7 % 4 * 3 /2

Operators used:

%, * and /

all three have same priority

so the expression will be calculated as per associvity i.e left to right

= 3 * 3 / 2

= 9 /2

= 4

***/**

7.

```
#include<stdio.h>
void main(){
        int a = 5;
```

```
        int b = ++a + a++ + --a;
        printf("Value of b is %d", b);
}
/*
   A. Value of x is 16
   B. Value of x is 21
   C. Value of x is 19
   D. none of the above
*/
/*
   Ans = C
   b = ++a + a++ + --a;                    // a = 5
    ++ has highest priority than + operator

     = a + a++ + --a                       // a = 6 return reference of it
     = a + temp + --a                      // a = 7 , temp = 6 , return temp
     = 7 + 6 + a                           // a = 6  , return reference
     = 7 + 6 + 6                           //then  addition will be done
     = 19
*/
```

8.

```
#include<stdio.h>
void main(){
        int a = 20;
        double b = 15.6;
        int c;
        c = a + b;
        printf("%d", c);
}
/*
   A. 35
   B. 36
   C. 35.6
   D. 30
*/
/*      Ans : A
        a  = 20      //type: int
        b = 15.6    // type double
        c = (garbage)   // type: int
        c = a+b
          = 20 + 15.6     //addition of two double or float values is always double value
          = 35.6
        c =  35          // but we are storing it in int type variable so it will truncate the mantisa i.e
digits after (.) and value is implicitly converted to the int

*/
```

9.

```
#include<stdio.h>
void main(){
```

```
        int a = 20, b = 15, c = 5;
        int d;
        d = a == (b + c);
        printf("%d", d);
}
/*
   A. 1
   B. 40
   C. 10
   D. 5
*/
```

/*

Ans : 1

expression d = a == (b+c)

here bracket has high priority

d = a == (15+5)

= a == 20

= 20 == 20 //comparision operator == : will evalutes to true if the both values are equal and return 1

= 1

***/**

10.

```
#include<stdio.h>
void  main () {
        int i = 10 ,j;
        j = ++ i;
        printf("%d, %d", i, j);
}
/*
a.      11 10
b.      11 11
c.      10 11
d.      10 10
*/
```

/*

Ans = b

i = 10

j = ++i

= i // i =11 and return reference of i

j = 11 // i chya box madhli value (incremented) J madhe copy keli jate

***/**