TRICKS IN C PROGRAMMING

1)BASICS TRICKS

1)

```
int main()
{
  float num1=2.5,num2=1.5;
  float num3=num1%num2;
  printf("num3=%d",num3);
 return 0;
}
```

OUTPUT: Compiler Error
Explanation:% operator can't be used with real numbers.if we want to get modules of two real numbers we can use fmod function like this.
Float num3=fmod(num1,num2);
To use fmod you must include math.h like this #include<math.h>

.....

2)
How would you round off a value from 1.66 to 2.0?

We can use ceil function in the header file <math .h> like this

```
float num1=1.66;
printf("%0.1f",ceil(num1));
```

.....

3) How would you round off a value from 1.66 to 1.0?

We can use ceil function in the header file <math .h> like this

```
float num1=1.66;
printf("%0.1f",floor(num1));
```

.....

4)By default real number is treated as?

By default if we have real number like this 1.5 this is treated as double number see below example:

```
float num1=1.66;
if(num1==1.66)
  printf("1.66 is float number");
else
  printf("1.66 is double number");
```

will print 1.66 is double number.

.....

5) int main()

```
{
  printf("%x", -1<<1);
  return 0;
}
```

Output is dependent on the compiler. For 32 bit compiler it would be fffffffe and for 16 bit it would be fffe

6)Consider the following two C lines:

```
int num1;
extern int num2;
```

First statement declare and define num1,but second statement only declare num2

……

7)

```
int num=20;
int main()
{
   int num=num;
   printf("%d", num);
   return 0;
}
```

First num is declared, then value is assigned to it.As soon as num is declared as a local variable,it hides the global variable num

……

8)

```
int main()
{
    extern int a;  //declaration
    printf("%d\n", a);
    return 0;
}
int a=20;  //definition
```

Output will be 20.

……

9)

```
extern int a; //declaration
int main()
{
    printf("%d\n", a);
    return 0;
}
int a=20; //definition
```

Output will be 20.

……

10)

```
extern int a;  //declaration
int main()
{
   int a=20; //definition
    printf("%d\n", a);
    return 0;
}
```

Output will be 20.

……

11)

```
extern int a;
int main()
{
    printf("%d\n", a);
    return 0;
}
```

Linker error:undefined refernce to a

12)

```
extern int a=5;
int main()
{
    int a=20;
    printf("%d\n", a);
    return 0;
}
```

Will give warning(a initialized and declared) but print 20

……

13)

```
extern int a=5;
int main()
{
    printf("%d\n", a);
    return 0;
}
```

Will give warning(a initialized and declared) but print 5

……

14)

```
extern int a;
int main()
{
    a=5;
    printf("%d\n", a);
    return 0;
}
```

Linker error: undefined refernce to a

……

15)

```
int main()
{
    extern int a;
    a=5;
    printf("%d\n", a);
    return 0;
}
```

Linker error:undefined reference to a
The statement `extern int a` specifies to the compiler that the memory for `'a'` is allocated in some other program and that address will be given to the current program at the time of linking. But linker finds that no other variable of name `'a'` is available in any other program with memory space allocated for it. Hence a linker error has occurred.

……

16)

```
int main()
{
  {
      int var = 10;
  }
  {
      printf("%d", var);

  }
```

Compiler error:var out of scope(undeclared in the scope of printf function)

……

16)

```
int main()
{
  int x = 032;
  printf("%d", x);
}
```

Will print 26?? 0 before number means that number is octal number(base 8) so the output will be
(2*8^0+3*8^1)=(2+24)=26

……

17)

```
int X=40;
int main()
{
    int X=20;
    printf("%d\n", X);
    return 0;
}
```

Whenever there is conflict between a local variable and global variable, the local variable gets priority.so the output will be 20.

……

18)

```
int main()
{
    int x = 10, y = 20, z = 5, i;
    i = x < y < z;
    printf("%d\n", i);
    return 0;
}
```

First condtion (x<y)->(10<20)? Fasle(0) and second condtion (0<z)->(0<5)? True(1) so the outputs is i=1.

……

19)

```
int main()
{
    int X=40;
    {
        int X=20;
        printf("%d ", X);
    }
    printf("%d\n", X);
    return 0;
}
```

Output:20 40 becouse of scope.

……

20)

```
int main()
{
    void v = 0;
    printf("%d", v);
    return 0;
}
```

Compiler error: Because of void is not data type so we can use it to define variable

……

21)which variable has the longest scope?

```
int a;
int main()
{
   int b;
 }
int c;
```

a has the longest scope. a is accessibl everywhere,b is limited to main() and c is accessible only after it's declaration

22)which of the following operation is incorrect?

```
a)int i=35; i=i%5;          //correct
b)short int j=255;j=j;     //correct
c)long int k=365L; k=k;   //correct
d)float a=3.14; a=a%3;   //incorrect(we can't use % for real numbers)
```

.....

23)

Which of the following is the correct usage of conditional operators used in C?

a) a>b ? c=30 : c=40;
b) a>b ? c=30;
c) max = a>b ? a>c?a:c:b>c?b:c
d) return (a>b)?(a:b)

Option A: assignment statements are always return in paranthesis in the case of conditional operator. It should be a>b? (c=30):(c=40);

Option B: it is syntatically wrong.

Option D: syntatically wrong, it should be return(a>b ? a:b);

Option C: it uses nested conditional operator, this is logic for finding greatest number out of three numbers

.....

24)what is unary operators ,binary operators, ternary operator?

Unary operator:operator is need one variable like(!,~,sizeof,++,--).
Binary operator:operator needs two variable like(Arithmetic operators except ++,--),(logical operators except !),(bitwise operators except ~).
Ternary operator(conditional operator):need 3 variable like a=a>1?b=5:c=2

.....

25)

```
int main()
{
    int i=-3, j=2, k=0, m;
    m = ++i && ++j && ++k;
    printf("%d, %d, %d, %d\n", i, j,
k, m);
    return 0;
}
```

-2 3 1 1 why?increment i so i will be -2, increment j so j will be 3 and increment k will be 1.
So m = 1&&-2&&1=1
So output will be i=-2, j=3, k=1, m=1

.....

26)

Assuming, integer is 2 byte, What will be the output of the program?

```
int main()
{
     printf("%x\n", -2<<2);
     return 0;
}
```

Firs we will convert -2 to binary we will use 2's complement
2->0000000000000010=0x0002
~2->1111111111111101=0xfffd
2's->1111111111111110=0xfffe=-2
-2<<2->1111111111111000=0xfff8.
So output will be 0xfff8.

.....

27)

```
int main()
{
    int i=-3, j=2, k=0, m;
    m = ++i || ++j && ++k;
    printf("%d, %d, %d, %d\n", i, j, k, m);
    return 0;
}
```

-2 2 1 1 why? We will use short circuit technique(optimization). We will increment i and true || anything=true.

……

28)

```
int main()
{
    int x=12, y=7, z;
    z = x!=4 || y == 2;
    printf("z=%d\n", z);
    return 0;
}
```

First condtion x!=4(true) so true||anything =true.
So the output will be (true=1).
Z=1.

……

29)

```
int main()
{
    int x=printf("%d",1234);
    return 0;
}
```

Output:Will print 12344
Explanation:printf will print 1234 and returns number of char so will print 12344

……

30)

```
int main()
{
    int x=printf("%d",1234);
    printf("%d",x);
    return 0;
}
```

Printf return number of characters so first will print 1234 and return number of (1234) ->4 so the output will be 12344.

……

31)

```
int  main()
{
   unsigned int x = -1;
   int y = ~0;
   if (x == y)
      printf("same");
   else
      printf("not same");
   return 0;
}
```

Same why? -1 and ~0 have the same pattern.
1=0000000000000001.
~1=1111111111111110.
-1=1111111111111111 -→1
0=0000000000000000
~0=1111111111111111 -→2
(1=2) so the output is same

……

32)

```
int main()
{
    int x=4, y, z;
    y = --x;
    z = x--;
    printf("%d, %d, %d\n", x, y, z);
    return 0;
}
```

First we will decrement x(pre decrement) so x=3 and y=3.
Second we assign value of x to z then decrement it(post decrement) so z=3 and x will become 2.
Output: 2 3 3

33)

```
int main()
{
     int i=3;
     i = i++;
    printf("%d\n", i);
    return 0;
}
```

Output:3 why?
First we put i++ in i so I is 3.

……

34)

```
int main()
{
     int i=3;
     i++;
    printf("%d\n", i);
    return 0;
}
```

Output:4 why?
We will increment I then print it.

……

35)

```
int main()
{
    int a=100, b=200, c;
    c = (a == 100 || b > 200);
    printf("c=%d\n", c);
    return 0;
}
```

Output:c=1.
Why?
First condition:c==10 true(1)→ true || anything=true(1). So the output will be c=1.

……

36)

```
int main()
{
    int x=55;
    printf("%d, %d, %d\n", x<=55, x=40, x>=10);
    return 0;
}
```

x<=55(true=1),x=40 will put 40 in x.
x>=10(true=1).
Output:1, 40, 1

……

37)

```
int main()
{
    int i=2;
    printf("%d, %d\n", ++i, ++i);
    return 0;
}
```

Output may vary from compiler to compiler.The order of evaluation of arguments passed to a function call is unspecified.Anyhow, we consider `++i, ++i` are Right-to-Left associativity. The output of the program is 4, 3.In TurboC, the output will be 4, 3.In GCC, the output will be 4, 4.

……

38)

```
int main()
{
    int k, num=30;
    k = (num>5 ? (num <=10 ? 100 : 200): 500);
    printf("%d\n", num);
    return 0;
}
```

Output:30.
num doesn't change.

39)

```
int main()
{
    int i=2;
    int j = i + (1, 2, 3, 4, 5);
    printf("%d\n", j);
    return 0;
}
```

Because, comma operator used in the expression `i (1, 2, 3, 4, 5)`. The comma operator has left-right associativity. The left operand is always evaluated first, and the result of evaluation is discarded before the right operand is evaluated. In this expression 5 is the right most operand, hence after evaluating expression (1, 2, 3, 4, 5) the result is 5, which on adding to i results into 7.

.....

40)

```
int main()
{
    char c=125;
    c=c+10;
    printf("%d",c);
    return 0;
}
```

Char range from -128 to 127 so when we add 10 to 125 the variable c will be 135. It is larger than the max range so overflow will occur, when overflow occur, subtract max size of variable(here is 256) from the value of variable(here 135) so output will be 135-256=-121.

.....

41)

```
int main()
{
    unsigned char c=125;
    c=c+10;
    printf("%d",c);
    return 0;
}
```

Unsigned char range from 0 to 255(overflow doesn't occur) so the output will be 135

.....

42)

Are the following two statement same?

a) a <= 20 ? (b = 30): (c = 30);

b) (a <=20) ? b : (c = 30);

No, the expressions 1 and 2 are not same.

`1)a <= 20 ? (b = 30) : (c = 30);` This statement can be rewritten as,

```
if(a <= 20)
{
    b = 30;
}
else
{
    c = 30;
}
```

2. `(a <=20) ? b : (c = 30);` This statement can be rewritten as,

```
if(a <= 20)
{
    //Nothing here
}
else
{
    c = 30;
}
```

.....

43)
```
int main()
{
    if(sizeof(int)>-1)
        printf("yes");
    else
        printf("no");
    return 0;
}
```
Sizeof(int)=4->0000000000000100
-1=1111111111111111(big number)
So output will print no.

……

44)
```
int main()
{
   float x=0.1;
   if(x==0.1)
    printf("if");
   else if(x==0.1f)
    printf("else if");
   else
    printf("else");
    return 0;
}
```
Real constants by default are double if we want to make it float we must write f after number so output will be else if.

……

45)
Suppose a c program has floating constant 1.414, what's the best way to convert this as "float" data types??

By default floating constant is of double data type. By suffixing it with f or F, it can be converted to float data type

……

46)
```
int main()
{
    int x;
    printf("%d", scanf("%d", &x));
    return 0;
}
```
In c scanf returns number of inputs it has successfully read.So the output will be 1.

……

47)
```
int main()
{
  printf("Mohamed %% gamal %% taha");
    return 0;
}
```
Output will be Mohamed % gamal % taha.
To print % we use %%

……

48)
```
int main()
{
    int a=10,b=15;
    printf("%d ",(a+1),(b=a+2));
    printf("%d",b);
    return 0;
}
```
Output: 11 12 why?
B will change in printf(b=a+2) so b will be a+2=10+2=12;

……

49)

```
int main()
{
    char x=127;
    printf("%d",1 + ++x);
    return 0;
}
```

```
Char range from -128 to 127.
++x->xwill be 128+1=129(overflow).
Output:129-256=-127
```

……

50)

How to find sum of two variables without using any operators?

```
int main()
{
    int x=4,y=10;
    int sum= printf("%*c%*c", x, '\r', y, '\r');
    printf("%d",sum);
    return 0;
}

Another solution bu using ++ and –
int main()
{
    int x=4,y=10;
   while(y)
   {
       x++;
       y--;
   }
   printf("%d",x);
    return 0;
}
```

……

51)

If x is an integer and it's size is not known at the moment, then the best way to initialize it to all 1's is x=…?(valeo technical exam)

a)~0.

b)!0.

c)0xffff.

```
Output?~0.
As ~0 will convert to 11111111111111111111111111111111.
Why not 0xffff? As this for int 2 bytes.if int 4 bytes will be wrong
and in the question size not known.
```

……

52)

The tool that combines object files or libraries and produces a single executable file is called?(Valeo technical exam)

a)compiler.

b)assembler.

c)preprocessor.

d)linker.

```
Linker
```

……

53)
Is it true a global variable may have several declarations,but only one definition?

Yes see the below example:

```
extern int x; // this is declaration
extern int x; // this is declaration
extern int x;  // this is declaration
int x=20;   //this is definition
int main()
{
   printf("%d",x);
    return 0;
}
```

……

54)

```
int main()
{
    int a=5,b=3,c=4;
   printf("a=%d ,b=%d",a,b,c);
    return 0;
}
```

a=5, b=3

……

55)

```
int main()
{
   int a=1;
   float b=1.3;
   double c;
   c=a+b;
   printf("%0.2lf",c);
    return 0;
}
```

2.30

……

56)

```
int main()
{
  int k=1;
printf("%d == 1 is" "%s\n", k,
k==1?"TRUE":"FALSE");
    return 0;
}
```

Output: 1==1 is TRUE Why?
For %d replace with k=1.
For conditional operator k==1(true) so will replace %s with TRUE.

……

57)

```
int main()
{
    float a=3.15529;
    printf("%0.1f\n", a);
    return 0;
}
```

Output:3.2.
Why not 3.1?
If number after 0.1 larger than or equal to 5 so icrease 0.1 to 0.2 if less 0.1 will print like this:

```
float a=3.14529;
printf("%0.1f\n", a);
```

will print 3.1 not 3.2

58)
In which numbering system can the binary number 1011011111000101 be easily converted to?

a) Decimal system.
b) Hexadecimal system.
c) Octal system.
d) No need to convert

Hexadecimal system. In embedded system we use hexadecimal system.

.....

59)
Which bitwise operator is suitable for turning off a particular bit in a number?

```
Bitwise & operator:
Num&=~(1<<bit).
```

.....

60)
Which bitwise operator is suitable for turning on a particular bit in a number?

```
Bitwise | operator:
Num|=1<<bit
```

.....

61)
Which bitwise operator is suitable for checking whether a particular bit is on or off?

```
Bitwise & operator:
((num>>bit)&1)
```

Example:

```
int main()
{
   int num=10;
   if((num>>1) &1) // check if bit '1' is 1 or 0
    printf("true");
   else
    printf("false");
    return 0;
}
```

.....

62)

```
int main()
{
    unsigned int a=0xffff;
    ~a;
    printf("%x\n", a);
    return 0;
}
```

~a has not effect so output will be ffff

.....

63)

```
int main()
{
    unsigned char i = 0x80;
    printf("%d\n",(i<<1));
    return 0;
}
```

Output:256
i=0x80(in hexa)=1000000(in binary).
i<<1 ->100000000=256(why not overflow occur?? As we need to cast it) see below example.If we do this i=i<<1.then print I output will be 0.

```
i=i<<1;
printf("%d",i); //print 0
```

64)

```
int main()
{
    unsigned char i = 0x80;
    printf("%d\n", (unsigned
char)(i<<1));
    return 0;
}
```

Output:0(overflow occur)
As range for unsigned char from 0 to 255 and output will be 256-256=0

……

65)

```
int main()
{
    int i=32, j=0x20, k, l, m;
    k=i|j;
    l=i&j;
    m=k^l;
    printf("%d, %d, %d, %d, %d\n",
i, j, k, l, m);
    return 0;
}
```

I assume one byte to be easy.
j=0x20(in hexa)=32(in decimal)=00100000(in binary)
K=i|j->00100000 | 00100000=00100000(32)
L=i&j->00100000&00100000=00100000(32)
M=k^l->00100000^00100000=00000000(0)
Output:32 32 32 32 0

……

66)

How to check for even or odd number by using bitwise operator?

```
int main()
{
    int num;
    printf("Enter a num: ");
    scanf("%d", &num);
    (num&1)?printf("odd"):printf("even");
    return 0;
}
```

Another method by using arithmetic operator:

```
((num%2)!=0)?printf("odd"):printf("even");
```

……

67)

How to print hello world without using semicolon:

1)Using if condition:

```
int main()
{
      if (printf("Geeks for Geeks")) {  }
}
```

2)Using while condition:

```
int main()
{
while (!printf( "Geeks for Geeks" )){}
}
```

3)Using switch case:

```
int main()
{
 switch (printf("Geeks for Geeks" )) {}
}
```

68)
How to print semicolon without using semicolon?

```
int main()
{
   // ASCII value of ; is 59
   if (printf("%c", 59))
   {
   }
}
```

……

69)

```
int main()
{
   char c='a';
 printf("%d %d %d
%d",sizeof(int),sizeof(c),sizeof('a'),sizeof("a"));
}
```

```
4 1 4 2 why?
Sizeof(int)=4 bytes.
Sizeof(char)=1 bytes;
Sizeof('a')=sizeof(int) as a will convert to it's ascii so will print 4.
Sizeof("a") this is string and we will take null in size(explain later)
will print 2.
```

……

70)

```
int main( )
{
   int x = 2, y = 5;
   if (x < y)
    return (x = x+y);
  else
    printf ("z1");
    printf("z2");
return 0; }
```

```
There is no compilation error but there
will no output  because function is returning a
value and if statement is true in this
case.
```

……

71)

Which of the following is the odd one out?

a)j=j+1.

b)j=+1;

c)j+=1;

d)j++;

```
Output:j=+1.
j=j+1 means increment j by 1.
j=+1 means j is positive 1.
j+=1 mean increment j by 1(like  j=j+1).
j++ means post increment
```

……

72)

```
int main(void)
{
 int x=256,y=2;
 int sum1=(char)x+3;
 int sum2 = (char)y+3;
 printf("sum1=%d sum2=%d",sum1,sum2);
      return 0; }
```

```
Output:sum1=3 sum2=5.
Explanation:we casting x to be char
and range of char from -128 to 127
so overflow will ocuur char(x) will
be 0 so sum1=0+3=0.
For y no overflow so sum2=2+3=5.
```

73)

```
int x=5;
int main()
{
    int x=3;
    printf("%d ",x);
    {
         x=4;
    }
    printf("%d",x);

    return 0;
}
```

Output:3 4
Explanation:when there is conflict between global and local, local get priority. So first x =3 and then reassign x with 4 so x become 4.

……

74)

```
int x=5;
int main()
{
    int x=3;
    printf("%d",x);
    {
         int x=4;
    }
    printf("%d",x);

    return 0;
}
```

Output:3 3
Explanation:when there is conflict between global and local, local get priority. So first x =3 and then we declare another variable in block scope so x still 3.

……

75)

```
int main()
{
   printf("%d",d++);

    return 0;
}
int d=10;
```

Output:compiler error
Explanation: d undeclared, to run successfully we must declare d before main like this: extern int d;

……

76)

```
int x=9;
int main()
{
    x=9;
    {
        int x=4; //variable will remove after bracket
    }
   printf("%d",x);
    return 0; }
```

Output:9
Explanation:we define x with 9 and then reassign it with 9 so x now is 9.

……

77)

```
int main()
{
    int x,y,z;
    z=scanf("%d %d",&x,&y);
    printf("%d",z);
    return 0;
}
```

Output:2
Explanation: scanf function returns number of inputs

78)
66)what is the function of the below code?

```
int main()
{
    float a=3.14;
    char *j;
    j = (char*)&a;
    printf("%d\n", *j);
    return 0;
}
```

It prints ASCII value of the binary number present in the first byte of a float variable a.

……

79)

```
int main()
{
     int x=512;
     char c=(char)x;
     printf("%d",c);
     return 0;
}
```

Output:0
Explanation: x is casting to be char so overflow will occur so c=512-256-256=0

……