=================== OPERATORS ========================

Evaluate the expression in a step by step way based on operator precedence and associativity

1)printf("%d\n",1==5==5);

Output:0

2) int i =0;

printf("%d %d",i,i++);

Output:1 0

3) int x=5;

printf("%d %d %d\n",x++,x++,x++);

Output:7 6 5

4) int x=2;

printf("%d ",++x++);

printf("%d\n",x++);

Output: printf statement shows error because we can’t assign pre increment and post increment .

5) int k=1;

printf("%d==1 is ""%s\n",k,k==1?"TRUE":"FALSE")

Output: 1==1 is TRUE

6) int i=5;

i=i++ - --i + ++i;

Output:6

7) int a=7;

a+=a+=a-=6;

printf("%d\n",a);

Output:4

8) int x=10,y=5,p,q;

p=x>9;

q=p||(x=5,y=10);

printf("%d %d %d\n",q,x,y);

Output:1 10 5

9) int x=2,y=1;

y+=x<<=2;

printf("%d %d\n",x,y);

Output:8 9

10) int x=2,y=4,z;

z=y++\*x++|y--;

printf("%d\n",z);

Output:13

11) int a=5,b=6,c=7,d;

d=a&=b&=c&&a;

printf("%d\n",d);

Output:0

12) int i=10;

i=(10<10)?(10>=10)?(10<=10)?1:2:3:4;

printf("%d\n",i);

Output:4

13) int a=10,b=20;

a=(a>5||b=6?40:50);

printf("%d\n",a);

Output:40

14) printf("%x\n",-1>>4);

printf("%x\n",-1<<4);

Output: ffffffff

fffffff0

15) int x=7;

x=(x<<=x%2);

printf("%d\n",x);

Output:14

16) int a=2,b=5,c=1;

printf("%d\n",(b>=a>=c?1:0));

Output:1

17) int a=5;

a=a-~a +1;

printf("%d\n",a);

Output:12

18) a=b=c=1;

x=--a||++b\*(3-1)/2&&b\*(--c/3);

printf("%d\n",x);

Output:0

19) a=10

b = -5

c = 2.5

printf("%d %d",sizeof(a) + sizeof((++b) \*c, b);

Output:8

20) int i=5;

i=i++ - i;

printf("%d",i);

Output: -1

========================================= PREPROCESSORS AND MACROS ===========================

1) What will be the output of the C program?

#include<stdio.h>

#define CONDITION(i)\

printf("preprocessor works\n");

int main()

{

CONDITION(0);

return 0;

}

**OUTPUT:**

preprocessor works

2) What will be the output of the C program?

#include<stdio.h>

# define loop while(true)

int main()

{

loop;

printf("preprocessor-aptitude");

return 0;

}

OUTPUT:error

3) What will be the output of the C program?

#include<stdio.h>

# define x –5 // we mentioned decrement macro to it

int main()

{

printf("%d",x); // it won’t be replaced by decrement

return 0;

}

OUTPUT:error

4) What will be the output of the C program ?

#include<stdio.h>

#define sqr(x) ++x \* ++x

int main()

{

int a = 3, z; // we define value 3 to ‘a’

z = ++a \* ++a; // here ‘a’ value increment and then multiplication

a -= 3;

printf("%d %d", sqr(a), z);

return 0;

}

OUTPUT: 16 25

5) What will be the output of the C program?

#include<stdio.h>

#define x 1+2

int main()

{

int i;

i = x \* x \* x;

printf("%d",i);

}

OUTPUT:7

6) What will be the output of the C program?

#include<stdio.h>

#define a =

int main()

{

int num a 6;

printf("%d",num);

return 0;

}

OUTPUT:6

7) What will be the output of the C program?

#include<stdio.h>

#define fun(x,y) x\*y

int main()

{

int x = 2, y = 1;

printf("%d",fun(x + 2, y - 1));

return 0;

}

OUTPUT:3

8) What will be the output of the C program?

#include<stdio.h>

int main()

{

char DATE[15] = "Current Date";

printf("%s\n", \_\_DATE\_\_ );

return 0;

}

OUTPUT: Aug 10 2021

9) What will be the output of the C program?

#include<stdio.h>

int main()

{

char TIME[15] = "Current Time";

printf("%s\n",\_\_TIME\_\_);

return 0;

}

OUTPUT: 06:37:00

10) What will be the output of the C program?

#include<stdio.h>

int main()

{

printf("Line :%d\n", \_\_LINE\_\_ );

return 0;

}

OUTPUT: Line :4

11) What will be the output of the C program?

#include<stdio.h>

#define preprocessor\_works(x, y) \

printf(#x " and " #y " are great!\n")

int main(void) {

preprocessor\_works(you, me);

return 0;

}

OUTPUT: you and me are great!

12) What will be the output of the C program?

#include<stdio.h>

# define puts "%s C preprocessor"

int main()

{

printf(puts, puts);

return 0;

}

OUTPUT: %s C preprocessor C preprocessor

13) What will be the output of the C program?

#include<stdio.h>

#define preprocessor(n) printf ("macro" #n " = %d", macro##n)

int main(void) {

int macro25 = 47;

preprocessor(25);

return 0;

}

OUTPUT: macro25 = 47

14) What will be the output of the C program?

#include<stdio.h>

#include<string.h>

#define MACRO(num) ++num

int main()

{

char \*ptr = "preprocessor";

int num =strlen(ptr);

printf("%s ", MACRO(ptr));

printf("%d", MACRO(num));

return 0;

}

OUTPUT: reprocessor 13

15) What will be the output of the C program?

#include<stdio.h>

#define i 10

int main()

{

#define i 20

printf("%d",i);

return 0;

}

OUTPUT:20

16) What will be the output of the C program?

#include<stdio.h>

#define clrscr() 50

int main()

{

clrscr();

printf("%d\n",clrscr());

return 0;

}

OUTPUT:50

17) What will be the output of the C program?

#include<stdio.h>

#define int char

main()

{

int i=5;

printf ("sizeof (i) =%d", sizeof (i));

}

OUTPUT: sizeof (i) =1

18) Write programs to understand the usage of below preprocessor directives.

#include, #if, #ifdef, #ifndef, #else, #elif, #endif, #define, #undef, #line, #error, and #pragma

#include :

#include <stdio.h>

#define MULTIPLY(a, b) a\*b

int main()

{

// The macro is expanded as 2 + 3 \* 3 + 5, not as 5\*8

printf("%d", MULTIPLY(2+3, 3+5));

return 0;

}

#define: The #define preprocessor directive is used to define constant or micro substitution. It can use any basic data type.

#include <stdio.h>

#define PI 3.14

main() {

   printf("%f",PI);

}

#undef **:** The #undef preprocessor directive is used to undefine the constant or macro defined by #define.

#include <stdio.h>

#define number 15

int square=number\*number;

#undef number

main() {

   printf("%d",square);

}

#ifdef :  The #ifdef preprocessor directive checks if macro is defined by #define. If yes, it executes the code otherwise #else code is executed, if present.

 #include <stdio.h>

#include <conio.h>

#define NOINPUT

void main() {

int a=0;

#ifdef NOINPUT

a=2;

#else

printf("Enter a:");

scanf("%d", &a);

#endif

*printf("Value of a: %d\n", a);*

*getch();*

*}*

*#ifndef:* The #ifndef preprocessor directive checks if macro is not defined by #define. If yes, it executes the code otherwise #else code is executed, if present.

#include <stdio.h>

#include <conio.h>

void main() {

int a=0;

#ifndef INPUT

a=2;

#else

printf("Enter a:");

scanf("%d", &a);

#endif

printf("Value of a: %d\n", a);

getch();

}

#if : The #if preprocessor directive evaluates the expression or condition. If condition is true, it executes the code otherwise #elseif or #else or #endif code is executed.

#include <stdio.h>

#include <conio.h>

#define NUMBER 0

void main() {

#if (NUMBER==0)

printf("Value of Number is: %d",NUMBER);

#endif

getch();

}

#else :

The #else preprocessor directive evaluates the expression or condition if condition of #if is false. It can be used with #if, #elif, #ifdef and #ifndef directives.

#include<stdio.h>

#include <conio.h>

#define NUMBER 1

void main() {

#if NUMBER==0

printf("Value of Number is: %d",NUMBER);

#else

print("Value of Number is non-zero");

#endif

getch();

}

#error :The #error preprocessor directive indicates error. The compiler gives fatal error if #error directive is found and skips further compilation process.

#include<stdio.h>

#include<math.h>

#ifndef \_\_MATH\_H

#error First include then compile

#else

void main(){

    float a;

    a=sqrt(7);

    printf("%f",a);

}

#pragma : The #pragma preprocessor directive is used to provide additional information to the compiler. The #pragma directive is used by the compiler to offer machine or operating-system feature.

#include<stdio.h>

#include<conio.h>

void func() ;

#pragma startup func

#pragma exit func

void main(){

printf("\nI am in main");

getch();

}

void func(){

printf("\nI am in func");

getch();

}

Elif :

#include <stdio.h>

#define YEARS\_OLD 12

int main()

{

#if YEARS\_OLD <= 10

printf("TechOnTheNet is a great resource.\n");

#elif YEARS\_OLD > 10

printf("TechOnTheNet is over %d years old.\n", YEARS\_OLD);

#endif

return 0;

===================================== TYPEDEF =========================================

1) In the following code, the P2 is Integer Pointer or Integer?

typedef int \*ptr;

ptr p1, p2;

Ans: Integer Pointer

2) In the following code what is 'P'?

typedef char\* charp;

const charp P;

Ans: pointer

3) What is x in the following program?

#include<stdio.h>

int main()

{

typedef char (\*(\*arrfptr[3])())[10];

arrfptr x;

return 0;

}

Ans:Array Pointer

4) What is apfArithmatics in the below statement?

typedef int (\*apfArithmatics[3])(int,int);

Ans: Integer Pointer

5) What is pf in the below statement?

typedef int (\*pf)(int);

Ans: Pointer to function

6) What do the following declarations mean?

typedef char \*pc; // Pointer to char

typedef pc fpc(); // Function with return type char

typedef fpc \*pfpc; // pointer to the fp()

typedef pfpc fpfpc(); // is function with pointer return

typedef fpfpc \*pfpfpc; // pointer to an above function

pfpfpc a[N]; // array of pointer of size N

7) Write few programs using typedef on structures and enums.

Example : structures

#include <string.h>

typedef struct Books {

char title[50];

char author[50];

char subject[100];

int book\_id;

} Book;

int main( ) {

Book book;

strcpy( book.title, "C Programming");

strcpy( book.author, "Nuha Ali");

strcpy( book.subject, "C Programming Tutorial");

book.book\_id = 6495407;

printf( "Book title : %s\n", book.title);

printf( "Book author : %s\n", book.author);

printf( "Book subject : %s\n", book.subject);

printf( "Book book\_id : %d\n", book.book\_id);

return 0;

}

Example : enum

typedef enum month billing\_cycle;

billing\_cycle month\_due;

typedef enum day {sun, mon, tue, wed, thu, fri, sat} day;

day today;

typedef struct date {

day day\_of\_week;

enum month month\_of\_year;

int day\_of\_month;

} date;

date due\_date;