ABOUT C

C is a general-purpose programming language created by Dennis Ritchie at the Bell Laboratories in 1972. It is a very popular language, despite being old. The main reason for its popularity is because it is a fundamental language in the field of computer science.

C is strongly associated with UNIX, as it was developed to write the UNIX operating system.

BASIC C STRUCTURE

structure.c > main()

```c
#include<stdio.h>

int main() {
    printf(" Hello World " );
    return 0 ;
}
```

ABOUT VARIABLE

a. Variable is the name of a memory location which stores some data.

b. Variables are case sensitive .

c. 1st character is alphabet or '_'

d. No comma/blank space .

e. No other symbol other than '_' .

aboutvariable.c > main()

```c
#include<stdio.h>

int main(){
    int age = 20 ;
    float real_number = 3.22 ;
    char special_chracter = '&' ;
    char alphabet = "a" ;
    return 0 ;
}
```

ABOUT CONSTANT

- Values that don't change(fixed) .
- There are a total of 3 types of constant .
- (a) Integer constant – 1,2,3,-1,-2 etc .
- (b) Real constant – 3.1416 , 2.4 ,4.55 etc.
- (c) Chracter constant ~ a , b, c ,d, & etc .

Cases of Constant

- For Integer constant ~ int ~ %d
- For Real constant ~ float ~%f
- For Chracter constant ~ char ~%c

constant.c > ...

```c
#include<stdio.h>

int main(){
    int integer = 20 ;
    float real_number = 3.14 ;
    char special_chracter = '*' ;
    printf(" The integer is %d" , integer );
    printf(" The real number is %f ", real_number );
    printf(" The special chracter is %c" , special_chracter);
    return 0 ;
}
```

COMMENT

comment.c > ...

```c
#include<stdio.h>
//single line comment : comment is not a part of code
/*multiple line comment : comment is not a part of code
It can be use for instruction
*/
int main(){

    return 0 ;
}
```

a. A line which is not a part of the code , which can be use for instruction .

b. Single Line comment ~ //

c. Multiple Line comment ~ /* */

BASIC SUM

sum.c > ...

```c
#include<stdio.h>

int main(){
    int number1 , number2;
    printf("Enter your number = ");
    scanf("%d" , &number1);
    printf("Enter another number = ");
    scanf("%d" , &number2);
    printf("your answer is = %d " , number1+number2);
    return 0 ;
}
```

a. Scanf ~ for taking input from user .

Question 1 : Write a code to calculate the area of a square .

problem1.c > ...

```c
//write a code to calculate the area of a square(side given)
#include<stdio.h>

int main(){
    int side;
    printf("Enter the mesure = ");
    scanf("%d" , &side );
    printf("Area of square is = %d " , side * side );
    return 0 ;
}
```

Question 2 : Write a code to calculate the area of a circle .

problem2.c > ...

```c
//Question= Write a code to calculate the area of a circle
#include<stdio.h>

int main(){
    float redious;
    printf("Enter the redious of the circle = ");
    scanf("%f" , &redious );
    printf("Area of circle is = %f" , 3.14*redious*redious );
    return 0 ;
}
```

Question 3 : Write a program to calculate perimeter of rectangle. Take sides from the user.

problem3.c > ...

```c
/*Question = Write a program to calculate perimeter of rectangle
Take sides, a & b, from the user*/
#include<stdio.h>

int main(){
    float length , width;
    printf("Enter length = ");
    scanf("%f" , &length);
    printf("Enter width = ");
    scanf("%f" , &width);
    printf("perimeter of rectangle is = %f" , 2*(length+width) );
    return 0 ;
}
```

Question 4 : Take a number(n) from user & output its cube(n*n*n).

problem4.c > ...

```c
//Question = Take a number(n) from user & output its cube(n*n*n).
#include<stdio.h>

int main(){
    float n ;
    printf("Enter a number for cube = ");
    scanf("%f" , &n );
    printf("Answer = %f" , n*n*n);
    return 0 ;
}
```

About Instruction :

- These are statements in a Program .
- There are a total 3 types of instruction .
  a. Type Declaration
  b. Arithmetic Instruction
  c. Control Instruction

TYPE DECLARATION

~ Always Declare the variable before using it .

- int a, b, c ;
  a = b = c = 1

First Declare the variable then use

- int a , b, c =1

Cannot declare variable and assign value same time

typedeclaration.c > ...

```c
#include<stdio.h>

int main(){
   // Valid
   int Age = 14 ;
   int oldAge = Age ;
   int newAge = oldAge + 1 ;
   printf("%d", newAge );

   //unvalid
   int age = 44;
   int newage = age , oldage ; // have to declare variable before using it .
   int oldage = 55;
   printf("%d", newage);

    //valid
    int a,b,c ;
    a=b=c=1;

    //unvalid
    int a = b = c = 1;  // cannot declare variable and assing value at the same time
    return 0 ;
}
```

ARITHMETIC INSTRUCTION

~ Cannot use multiple variable on the left side .

- a = b + c ~ Correct
- b + c = a ~ Wrong
- Module Operator (%) ~ Return the reminder .

  ~ int 3 % 2 = 1

- Type Conversion ~ its provide the value which takes more space .
  a. int op int ~ int
  b. Int op float ~ float
  c. Float op float ~ float

- Operator Precedence / associavity ~ *,/,% ~ +,- ~ =

> In C, operator precedence determines the order in which operators are evaluated in an expression. Operators with higher precedence are evaluated first. If operators have the same precedence, their associativity (either left-to-right or right-to-left) comes into play.

associavity.c > ...

```c
#include<stdio.h>

int main(){
    printf("answer = %d " , 2*(2+2)*2/2 );
    return 0 ;
}
//associative order as priority  and if same left to right .
```

ABOUT OPERATOR

- They mainly Gives True or False value . 1 = True , 0 = False .
- There are many types of operator in C . Most used are ~
  a. Relation Operator
  b. Logical Operator
  c. Assignment Operator
  d. Bitwise Operator
  e. Arithmetic Operator
  f. Ternary Operator

RELATIONAL OPERATOR

- **<** (Less than)
- **>** (Greater than)
- **<=** (Less than or equal to)
- **>=** (Greater than or equal to)
- **==** (Equal to)
- **!=** (Not equal to)

relationalOperator.c > ...

```c
#include<stdio.h>

int main(){
    printf("%d \n" , 4==4);
    printf("%d \n" , 4>3);
    printf("%d \n" , 4>=3);
    printf("%d \n" , 4<3);
    printf("%d \n" , 4<=3);
    printf("%d \n" , 4!=3);
     return 0 ;
}
```

LOGICAL OPERATOR

- **&&** ~~ (Logical AND) ~ Output will be True(1) if both condition is true .
- || **~~** (Logical OR) ~~ Output will be True(1) if one condition is true .
- **!** ~~ (Logical NOT) ~~ Depends .

LogicalOperator.c > main()

```c
#include<stdio.h>

int main(){
    printf("%d \n" , 4==4 && 4<3);
     printf("%d \n" , 4==4 && 4>3);
    printf("%d \n" , 4==4 || 4<3);
    printf("%d \n" , !(4==3) && !(4<3) );
     printf("%d \n" , !(4==3) && !(4>3) );
     return 0 ;
}
```

ASSIGNMENT OPERATOR

~ Assignment operator is used to assign value , variable and function to another function

- **=** (Assignment) ~ Assigns the value on the right to the variable on the left
- **+=** (Addition assignment) ~ x += 5; // Equivalent to x = x + 5;
- **-=** (Subtraction assignment) ~ x -= 5; // Equivalent to x = x - 3;
- ***=** (Multiplication assignment) ~ x *= 5; // Equivalent to x = x * 2;
- **/=** (Division assignment) ~ x /= 5; // Equivalent to x = x / 4;
- **%=** (Modulus assignment ~ x %= 5; // Equivalent to x = x % 3;

assignmentOperator.c > ...

```c
#include<stdio.h>

int main(){
    int num1 = 5 ;
    num1+=5 ;
    printf(" answer = %d \n" , num1);
    num1-=5 ;
    printf(" answer = %d \n" , num1);
    num1*=5 ;
    printf(" answer = %d \n" , num1);
    num1/=5 ;
    printf(" answer = %d \n" , num1);
    return 0 ;
}
```

Question 5 : Write a programe to detect two digit number .

problem5.c > main()

```c
// Question : write a programe to decect
#include<stdio.h>

int main(){
    int num1 ;
    printf(" Enter your Number = ");
    scanf("%d" ,&num1);
    printf("%d" , num1>9 && num1<100);
    return 0 ;
}
```

Question 6 : Write a programe to calculate average of 3 number

problem6.c > ...

```c
//question : write a programme to calculate the averc
#include<stdio.h>

int main(){
    float num1, num2, num3 ;
    printf("Enter first number : ");
    scanf("%f", &num1);
    printf("Enter Second number = ");
    scanf("%f" , &num2);
    printf("Enter third number = ");
    scanf("%f" , &num3);
    printf("Average is = %f" , (num1+num2+num3)/3);
    return 0 ;
}
```

problem6short.c > main()

```c
#include<stdio.h>

int main(){
    float num1 ,num2 ,num3 ;
    printf("enter 3 of your number( use space) = ");
    scanf("%f%f%f" , &num1 , &num2 ,&num3);
    printf("average is = %f" , (num1+num2+num3)/3 );
    return 0 ;
}
```

Question 7 : Write a programe to find the smallest number between them .

problem7.c > ...

```c
// write a code to find the smallest number
#include<stdio.h>

int main(){
    int num1,num2 ;
    printf(" Enter your number = ");
    scanf("%d" , &num1);
    printf("enter another number = ");
    scanf("%d" , &num2);
    num1>num2 && printf("%d" , num2) || printf("%d" , num1);

    /*(&&) operator to check if num1 is less than num2.
     If that condition is true, it proceeds to execute the first printf statement,*/

     return 0 ;
}
```

Question 8 : Write a program to check if given character is digit or not. .

problem8.c > main()

```c
//Write a program to check if given character is digit or not
#include <stdio.h>

int main() {
    char character;

    // Input a character
    printf("Enter a character: ");
    scanf(" %c", &character);

    // Check if the character is a digit using relational and logical operators
    ((character >= '0') && (character <= '9')) && printf("%c is a digit.\n", character) || printf("%c is not a digit.\n",character);
    // by using ternary

     //(character>='0' && character<='9') ? printf("%c is a digit", character) : printf("%c is not a digit" , character);

    return 0;
}
```

Conditionals : ~ Conditional Statement are programming constract that allow a programme to execute different block of code based on weather a certain condition in True or False .

~~Control flow is managed through conditional statements (**if**, **else**, **switch**)~~

IF ,ELSE

~~Basic if Structure ~

```
if (condition1) {
// Code to be executed if condition1 is true
} else if (condition2) {
// Code to be executed if condition2 is true
} else if (condition3) {
// Code to be executed if condition3 is true
} else {
// Code to be executed if none of the conditions are true
}
```

Example :

```c
#include<stdio.h>

int main(){
    int runs ;
    printf("Enter your run = ");
    scanf("%d" , &runs);

    if (runs>=50 && runs<100) {
        printf("You made a half century !");
    }
    else if (runs>=100 && runs<=150) {
     printf("Woah ! You made a century ");
    }
    else if(runs>150){
        printf("Too much");
    }

    else{
        printf("you lose ");
    }
```

TERNARY

~provides a concise way to express a conditional statement.

~Basic Structure is ~

~ **condition ? do something if_true : do something if_false;**

- **Do something_if_true**: If the condition is true, the value or expression following the **?** is returned.
- **Do something _if_false**: If the condition is false, the value or expression following the **:** is returned.

Example :

```c
#include <stdio.h>

int main() {
    int num;

    // Input a number
    printf("Enter a number: ");
    scanf("%d", &num);

    // Using the ternary operator to check if the number is even or odd
    printf("The number is %s.\n", (num % 2 == 0) ? "even" : "odd");

    return 0;
}
```

SWITCH

~~Basic Structure~

```
switch (expression) {
case constant 1:
   // Code to be executed if expression matches constant1
   Break ;
case constant2:
   // Code to be executed if expression matches constant2
   break;
   default:
   // Code to be executed if no cases match
}
```

Example : Switch

switch.c

```c
#include<stdio.h>

int main(){
    int lucky_number;
    printf(" Write a number for lottery(1-7) =  ");
    scanf("%d" , &lucky_number);

    switch (lucky_number){
        case 1 : printf("ooPss ! Better luck next time ");
                 break;
        case 2 : printf("ooPss ! Better luck next time ");
                 break;
        case 3 : printf("ooPss ! Better luck next time ");
                 break;
        case 4 : printf("ooPss ! Better luck next time ");
                 break;
        case 5 : printf("ooPss ! Better luck next time ");
                 break;
        case 6 : printf("woahhhh ! You wonnn .  ");
                 break;
        case 7 : printf("ooPss ! Better luck next time ");
                 break;
        default:
        printf(" invalid card ");
    }
    return 0 ;
}
```

Example : Switch for Character

switchForChracter.c

```c
#include<stdio.h>

int main(){
    char lucky_alphabet;
    printf(" Enter a alphabet fo lottery(a-z) =  ");
    scanf("%c" , &lucky_alphabet);

    switch (lucky_alphabet){
        case 'c' : printf("ooPss ! Better luck next time ");
                   break;
        case 'b' : printf("ooPss ! Better luck next time ");
                   break;
        case 'a': printf("ooPss ! Better luck next time ");
                   break;
        case 'e' : printf("ooPss ! Better luck next time ");
                   break;
        case 'f' : printf("ooPss ! Better luck next time ");
                   break;
        case 'g' : printf("woahhhh ! You wonnn .  ");
                   break;
        case 'h' : printf("ooPss ! Better luck next time ");
                   break;
        default:
        printf(" invalid card ");
    }
    return 0 ;
```

Question 9 : Write a program to cheak if a student pass or not . pass = >30 fail = <30

problem9.c > main()

```c
//write a program to cheak if a student pass or not . pass = >30 fail = <30
#include<stdio.h>

int main(){
    int marks;
    printf("Enter Your marks = ");
    scanf("%d" , &marks);
    if(marks<30){
        printf("You Fail !");
    }
    else if ( marks >= 30 && marks <=100) {
     printf(" You Passed !! ");
    }
    else{
        printf("invalid number");
    }
    return 0 ;
}
```

Question 10 : Write a program to give student grade . marks<30 is c , 30<=marks<70 is B , 70<=marks<90 is A , 90<=marks<=100 is A

```c
#include<stdio.h>

int main(){
    int marks;
    printf("Enter your marks =");
    scanf("%d", &marks);
    if(marks<30) {
        printf("You got C !");
    }
    else if (marks>=30 && marks<70){
        printf("You got B !");
    }
    else if (marks>=70 && marks<90){
        printf("You got A !");
    }
    else if (marks>= 90 && marks <= 100){
        printf("Woah ! You got A++");
    }
    else {
        printf("Invalid number ");
    }
    return 0 ;
}
```

Question 11 : Write a program to to find if a chracter entered by user is upper or lower case

problem11.c > main()

```c
//Question : write a program to to find if a chracter entered by u
#include<stdio.h>

int main(){
    char chracter ;
    printf("Enter Your Chracter = ");
    scanf("%c" , &chracter);
    //When comparing characters, you should use single quotes (')
    if (chracter >= 'A' && chracter <= 'Z'){
    printf("Its upper case");
    }
    else if (chracter >='a' && chracter <= 'z'){
        printf("Its a lower Case ");
    }
    else{
        printf("invalid");
    }
    return 0 ;
}
```

Question 12 : Write a program Check if the number is a natural number

problem12.c > main()

```c
//Question : Check if the number is a natural number
#include<stdio.h>

int main(){
    int num ;
    printf("Enter a number = ");
    scanf("%d" , &num);
    if(num > 0){
        printf("%d is a natural number .\n" ,num );
    }else{
        printf("%d is not a natural number . \n" ,num );
    }
     return 0 ;
}
```

About Loops ~~ Used to repeat some part of program .

~~ A total 3 types of Loops~~ For Loops

~~ While Loops

~~ Do-While Loops

FOR LOOPS

~~Basic Structure

```
for (initialization; condition; update) {
// Code to be executed while the condition is true
}
```

Example :

```c
#include<stdio.h>

int main(){
    int i;
    for(int i = 1; i<=5 ; i=i+1){
    printf("Hello World \n" , i);
    }
    return 0 ;
}
```

Question 13 : Print the Numbers from 0 to 10 .

```c
//Question: print a number from 0 to 10
#include<stdio.h>

int main(){
    for(int i = 0 ; i<=10 ; i = i+1){
        printf("%d \n" , i);
    }
    return 0 ;
}
```

Increasement Operator ~~ i=i+1~~i++(use, then increase) / ++i(increase, then use)

```c
// increasementOperator.c > main()
#include<stdio.h>

int main(){
    int i =0;
    printf("%d\n" , i++); //use, then increase
    printf("%d\n" , i);

    printf("%d\n",++i);
    printf("%d\n" , i);//increase , then use
    return 0;
}
```

```c
// characterLoop.c > main()
#include<stdio.h>

int main(){
    for(char ch='a'; ch<='z'; ch++){
        printf("%c \n" , ch);
    }
    return 0;
}
```

WHILE LOOPS

~~Basic Structure~

```
while (condition) {
// Code to be executed while the condition is true
}
```

```c
// whileLoop.c > main()
#include<stdio.h>

int main(){
    int i = 1 ;
    while(i<=5){
        printf("Hello World\n" , i);
        i++;
    }
    return 0 ;
}
```

Question 14 : Print the Numbers from 0 to n , n is given by user

```c
// problem14.c > main()
#include<stdio.h>

int main(){
    int n ;
    printf("Enter your number = ");
    scanf("%d", &n);

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

DO WHILE LOOPS

~~Basic structure~~

```
do {
// Code to be executed at least once, and then repeatedly while the condition is true
} while (condition);
```

Example :

DoWhileLoop.c > main()

```c
#include<stdio.h>

int main(){
    int i = 10;
    do{
        printf("%d\n" , i);
        i--;
    } while(i>=1);
    return 0 ;
}
```

Question 15 : Print the sum of first n Natutal Number ;

problem15.c > ...

```c
#include<stdio.h>

int main(){
    int n;
    printf("Enter your number = ");
    scanf("%d" , &n);

    int sum = 0;
    for(int i=1; i<=n; i++){
        sum = sum + i;
    }

    printf("sum is = %d" , sum);
    return 0 ;
}
```

Question 16 : Print the table of a number input by the user

problem16.c > ...

```c
#include<stdio.h>

int main(){
    int n;
    printf("Enter your number =");
    scanf("%d" , &n);

    for(int i=1; i<=10; i++){
        printf("%d\n" , n*i);
    }
    return 0 ;
}
```

Question 17 : Keep taking numbers as input from user untill user enter an odd number

problem17.c > main()

```c
#include<stdio.h>

int main(){
    int num;
    printf(" Enter your number = ");
    scanf("%d", &num);
    while (num)
    {
        if (num %2 !=0 )
        {
            printf(" odd number decected .");
            break;
        }else{
            printf("%d\n" , num);
        }
        printf("Enter number again = ");
        scanf("%d", &num);
    }
    return 0;
}
```

About Break and continue :

Break : The break statement is used to terminate the execution of a loop prematurely, before the loop condition is false .

aboutbreak.c > main()

```c
#include<stdio.h>

int main(){
    int i = 0;
    for(int i = 0 ; i<=5; i++){
        if(i==3){
            break;
        }
        printf("%d\n" , i);
    }

    return 0 ;
}
```

```
PS D:\C- Apna Colleage> gcc aboutbreak.c
PS D:\C- Apna Colleage> ./a.exe
0
1
2
PS D:\C- Apna Colleage>
```

Continue : The continue statement is used to skip the rest of the code inside the loop for the current iteration and move on to the next iteration of the loop.

aboutcontinue.c > main()

```c
#include<stdio.h>

int main(){
    for(int i= 0 ; i<=5 ;i++){
        if(i==3){
            continue;
        }
        printf("%d\n" , i);
    }
}
```

```
PS D:\C- Apna Colleage> gcc aboutcontinue.c
PS D:\C- Apna Colleage> ./a.exe
0
1
2
4
5
```

Question 18 : Keep taking numbers as input from user untill user enter a number which is multiple of 7

problem18.c > main()

```c
#include<stdio.h>

int main(){
   int num;
   printf("Enter your number = ");
   scanf("%d" , &num);
   while(num){
       if(num % 7 == 0){
           printf("Multiple of 7 detected ! ");
           break;
       }else{
           printf("Nope ! wrong \n");
           printf("Enter your number again = ");
           scanf("%d" , &num);
       }
   }
   return 0 ;
}
```

Question 19 : Print all the odd number from 5 to 50

problem19.c > main()

```c
#include<stdio.h>

int main(){
    int num = 5;
    while(num<=50){
        if( num%2 != 0){
        printf("%d\n" , num);
        }
        num++;
    }
    return 0;
}
```

.

Question 20 : Print the factorial of number provide by user

problem20.c > main()

```c
#include<stdio.h>

int main(){
    int num;
    printf("Enter your number =");
    scanf("%d" , &num);

    int fact = 1;
    for(int i =1; i<=num ; i++){
        fact= fact*i;
    }
    printf("factorial is = %d" , fact);
    return 0 ;
}
```

Question 21 : Print reverse table of a number input by the user

problem21.c > main()

```c
#include<stdio.h>

int main(){
    int num;
    printf("Enter your number = ");
    scanf("%d", &num);
    for(int i=10; i>=1; i--){
        printf("%d\n" , num*i);
    }
    return 0;
}
```

Question 22 : Calculate the sum of all numbers between 5 and 50,include 5 &50

problem22.c > main()

```c
#include<stdio.h>

int main(){
    int sum =0;
    for(int i; i<=50; i++ ){
       sum = sum + i;
    }
    printf("Total sum is = %d", sum);
    return 0;
}
```

Question 23: Print the pattern

```
* * * * *
* * * * *
* * * * *
* * * * *
```

problem23.c > ...

```c
#include<stdio.h>

int main(){
    int rows = 4;
    int coloum = 5;
    for(int i = 1; i<=rows; i++ ){
        for(int j=1; j<=coloum; j++){
            printf("* ");
        }
        printf("\n");
    }


    return 0;
}
```

Question 24: Write a program to detect prime number .

problem24.c > main()

```c
#include<stdio.h>

int main(){
    int num;
    printf("Enter your number = ");
    scanf("%d", &num);
    int prime =1;
    if(num<=1){
    prime=0;
    }else{
        for(int i= 2; i<num ;i++){
            if(num % i == 0){
                prime = 0;
                break;
            }
        }
    }
    if(prime){
        printf("%d is a prime number " , num);
    }else{
        printf("%d is not a prime number ", num);
    }
    return 0;
}
```

In a range is also the same with some correction.

ABOUT FUNCTION

Block of code that performs particular task . Functions provide a way to organize and modularize code, making it easier to read, understand, and maintain. Functions in C are declared with a return type, a name, and a parameter list . **Basic structure ~~**

```
// Function declaration
return_type function_name(parameter_type parameter_name) {
    // Code to perform the task
    // Return statement (if the function has a return type)
    return value;
}
int main() {
    // Function call
    return_type result = function_name(argument_value);
    // Rest of the program
    return 0;
}
```

Void

Int/float

Return_type **function_name**

```c
#include<stdio.h>
//Function declaration
void greeting(){
    printf("Hello");
} // no return value cause , voild has no return value

int main(){
    //function call
    greeting();
    return 0 ;
}
```

Call

Question 25: Write two Function _one to print Hello & other one to print GoodBye .

problem25.c > SayBye()

```c
#include<stdio.h>

void SayHello(){
    printf(" Hello!\n");
}
void SayBye(){
    printf("Good Bye !\n ");
}

int main(){
    SayHello();
    SayBye();
    return 0;
}
```

Question 26: Write a function that prints Namaste if user is Indian & print Bonjour if France .

problem26.c > main()

```c
#include<stdio.h>


void indian(){
    printf("Namastee");
}
void France(){
    printf("Bonjour");
}
int main(){
    printf("Enter your nationality (I)Indian , (F) France : ");
    char natioality;
    scanf("%c", &natioality);
    if(natioality=='I'){
        indian();
    }else if(natioality=='F'){
        France();
    }else{
       printf("Invalid key !");
    }
    return 0;
}
```

Some more info about Function :

- Execution always starts from main Properties
- A function gets called directly or indirectly from main
- There can be multiple functions in a program

Function Type

- **Library Function~~** Buil-function in C (printf, Scanf)
- **User Defined~~** Function created and defined by user

Question 27: Use Library function to calculate the square of a number given by user .

problem27.c > main()

```c
#include<stdio.h>
#include<math.h>
int square(int n){
    int result = pow( n ,2);
    return result ;
}

int main(){
    printf("Enter your number = ");
    int num;
    scanf("%d" , &num);
    int Final_result = square(num);
    printf("Area of square is = %d " , Final_result);
    return 0;
}
```

Question 28: Write a function to calculate area of a square, a circle & a rectangle .

problem28.c > main()

```c
#include<stdio.h>

float circle(float radious){
    float result = 3.14*radious*radious;
    return result;
}

float square(float side){
    float result2 = side*side;
    return result2;
}

float rectangle(float width , float length){
    float result3 = 2*(width+length);
    return result3;
}

int main(){
    int choise;
    printf("Select Your shape : \n");
    printf(" 1.Circle\n 2.Square\n 3.Rectangle\n");
    printf("Enter number(1-3) : ");
    scanf("%d",&choise);

        switch(choise){
        case 1 : printf("Enter radious of circle = ");
                 float radious;
                 scanf("%f" , &radious);
                 float Final_result = circle(radious);
                 printf("Area of circle is = %f" , Final_result);
        break;

        case 2 : printf("Enter side of square = ");
                 float side;
                 scanf("%f" , &side);
                 float Final_result2 = square(side);
                 printf("Area of Circle is = %f" , Final_result2);
        break;

        case 3 :printf("Enter lenth of the rectangle = ");
                 float  lenth , width ;
                 scanf("%f" , &lenth);
                 printf("Enter width of the rectangle = ");
                 scanf("%f" , &width);
                 float Final_result3 = rectangle(lenth , width);
                 printf("Area of rectangle is = %f" , Final_result3);
        break;

        default :
        printf("Invalid Choise !! ");
    }
    return 0 ;
}
```

Some more info about Function : → **Parameter** → **Return Value**

- functions can take value & give some value .
- Function can only return one value at a time
- Changes to parameters in function don't change the values in calling function. Because a copy of argument is passed to the function .

Argument :	**Parameter:**
• Values that are past in function call	a. Values in function defenation
• Used to send value	b. Used to receive value
• Its called Actuall parameter	c. Its called formal Parameters

ABOUT RECURSION

When a function calls itself, it's called recursion .

- Anything that can be done with Iteration, can be done with recursion and vice-versa.
- Recursion can sometimes give the most simple solution.
- Iteration has infinite loop & Recursion has stack overflow
- Base Case is the condition which stops recursion.

Basic structure :

recursion.c > main()

```c
#include<stdio.h>

void recursion(int n){//create function
    if(n==0){//base case where recursion stop.
    return;
    }else{
    printf("hello World\n");
    recursion(n-1);//the rule of recursion
}
}
 int main(){
    recursion(5);
 }
```

Question 29: Write a programme to calculate sum of first n natural number (recursion)

problem29.c > main()

```c
#include<stdio.h>

int sum(int n){
if(n==1){
    return 1;
}else{
    int sumNm=sum(n-1);
    int sum = sumNm+n;
    return sum;
}
}
int main(){
    printf("Enter your number = ");
    int num;
    scanf("%d",&num );
    int result = sum(num);
    printf("Your sum is = %d" , result);
}
```

Question 30: Write a programme to calculate factorial of n (recursion)

problem30.c > main()

```c
#include<stdio.h>

int factorial(int n){
    if(n==1){
        return 1;
    }else{
        int fact = factorial(n-1);
        int result = fact*n;
        return result;
    }
}

int main(){
    printf("Enter your number = ");
    int num;
    scanf("%d", &num);
    int result= factorial(num);
    printf("Factorial is = %d" , result);
    return 0;
}
```

Question 31: Write a programme to convert celcious to fahrenheit

problem31.c > main()

```c
#include<stdio.h>

float convertor(float n){
    float far = n * (1.8) + 32;
    return far;
}

int main(){
    printf("Enter your temperature(C) : ");
    float celcious;
    scanf("%f",&celcious);
    float Final_result = convertor(celcious);
    printf("Temperature is = %f " ,Final_result);
    return 0 ;
}
```

Question 32: Write a programme to calculate percentage of 3 subject number

problem32.c > main()

```c
#include<stdio.h>

float percentange(float a , float b ,float c){
    float result = (a+b+c)/300 * 100;
    return result;
}

int main(){
    printf("Enter Your number of subject 1 = ");
    float num1,num2,num3;
    scanf("%f", &num1);
    printf("Enter your number of subject 2 = ");
    scanf("%f", &num2);
    printf("Enter your number of subject 3 = ");
    scanf("%f", &num3);

    float Final_result =percentange(num1,num2,num3);
    printf("Your percentange is = %f" , Final_result);
    return 0;
}
```

Question 33: Write a programme to create Fibonacci Sequence

problem33.c > main()

```c
#include<stdio.h>

int fibo(int n){
    if(n==0){
        return 0;
    }if(n==1){
        return 1;
    }else{
        int FibNm = fibo(n-1);
        int FibNm2 = fibo(n-2);
        int result = FibNm+FibNm2;
        return result;
    }
}
int main(){
    printf("Enter 'n'th Number = ");
    int n;
    scanf("%d",&n);
    int Final_result = fibo(n);
    printf("Number is = %d", Final_result);
}
```

Question 34: Write a programme to find sum of digit of a number

problem34.c > main()

```c
#include<stdio.h>

int digit(int n){
    if(n<9){
        return n; //%10 ~~ 3 // n/10~~12
    }
    int result = n%10 + digit(n/10);
    return result;
}

int main(){
    printf("Enter your Number = ");
    int num;
    scanf("%d", &num);
    int Final_result = digit(num);
    printf("sum of the digits are = %d", Final_result);
}
```

Question 35: Write a programme to find square root of a number

problem35.c > main()

```c
#include<stdio.h>
#include<math.h>

float root(float n){
    float result = sqrt(n);
    return result;
}

float main(){
    printf("Enter your number = ");
    float num;
    scanf("%f",&num);
    float Final_result = root(num);
    if(num<=0){
        printf("Invalid !!");
    }else{
        printf("square root is = %f", Final_result);
    }
}
```

Question 36: Write a code to detect it Hot or Cold depending on temperature user enter

problem36.c > main()

```c
#include<stdio.h>

float temp(int n){
    if(n<=25 && n>0){
        printf("Its Cold ");
    }else if(n>25){
        printf("Its Hot");
    }else{
        printf("Invalid key!!");
    }
}

int main(){
    printf("Enter temperature : ");
    int num;
    scanf("%d", num);
    temp(num);
    return 0;
}
```

ABOUT POINTER

Pointers in C are variables that store the memory address of another variable.

BASIC STRUCTURE:

int x = 10; // An integer variable

int *ptr; // Declaration of a pointer variable

ptr = &x; // Initialization of the pointer with the address of x

aboutPointer.c > main()

```c
#include<stdio.h>

int main(){
    int x = 10;
    int*ptr ;
    ptr =&x;

    printf("Value of x is : %d\n" ,x);
    printf("Address of x is  :%d\n ",&x);
    printf("Value enter in ptr is  :%d\n" , ptr);
    printf("Value at the address stored in ptr :%d\n " , *ptr);
    return 0;
}
```

- **(& Operator)**: To get the memory address of a variable.
- **(* Operator):** To access the value stored at the memory address pointed to by a pointer

Question 37: Find Output

problem37.c > main()

```c
int main(){
    int*ptr;
    int x;

    ptr = &x;
    *ptr = 0 ;

    printf("x = %d\n" , x); // ans : 0
    printf("*ptr = %d\n" , *ptr); //ans : 0

    *ptr += 5;
    printf(" x = %d\n" , x); // ans : 5
    printf(" *ptr = %d\n" , *ptr); // ans :5

    (*ptr)++;
    printf(" x = %d\n" , x); // ans : 6
    printf(" *ptr = %d\n" , *ptr); // ans : 6
    return 0;
}
```

ABOUT POINTER TO POINTER : A pointer to a pointer (double pointer) in C is a variable that holds the memory address of another pointer.

BASIC STRUCTURE

int x = 10;

int *ptr1 = &x; // Pointer to an integer

int **ptr2 = &ptr1; // Pointer to a pointer to an integer

pointerTopointer.c > main()

```c
#include <stdio.h>

int main() {
    int x = 10;
    int *ptr1 = &x;       // Pointer to an integer
    int **ptr2 = &ptr1;   // Pointer to a pointer to an integer

    printf("Value of x: %d\n", x);
    printf("Value at address stored in ptr1: %d\n", *ptr1);
    printf("Value at address stored in ptr2: %d\n", **ptr2);

    // Modifying the value of x using ptr1
    *ptr1 = 20;
    printf("Modified value of x using ptr1: %d\n", x);
    printf("Value at address stored in ptr2 after modification: %d\n", **ptr2);

    // Modifying the value of x using ptr2
    **ptr2 = 30;
    printf("Modified value of x using ptr2: %d\n", x);
    printf("Value at address stored in ptr1 after modification: %d\n", *ptr1);

    return 0;
}
```

POINTER IN FUNCTION CALL

CALL BY VALUE (We pass the value of Variable)

CALL BY REFERANCE (We pass the adress of the Variab;e)

CALL BY VALUE : In call by value, the actual values of the arguments are passed to the function. The function receives a copy of the values, and any modifications made to the parameters inside the function do not affect the original values in the calling function.

callByvalue.c > main()

```c
#include<stdio.h>

int square(int n){
    n = n*n;
    printf("square = %d\n", n);
}

int main(){
    int number = 10;
    square(number);
    printf("square = %d\n" , number);
}
```

CALL BY REFERANCE : In call by reference, the memory address (reference) of the actual variables is passed to the function. The function can directly manipulate the values at those addresses, and any changes made inside the function are reflected in the calling function.

callByrefarence.c > square(int *)

```c
#include<stdio.h>

int square(int *n){
    *n = (*n) * (*n);
    printf("square = %d\n" , *n);
}
int main(){
    int number = 10;
    square(&number);
    printf("square = %d\n" , number);
}
```

Question 38: SWAP 2 number , a & b

problem38.c > ...

```c
#include<stdio.h>

int swap(int *a, int *b){
    int c = *a;
    *a = *b;
    *b = c;
}

int main(){
    int a = 2 , b=3;
    swap(&a ,&b);
    printf("a = %d\n" , a);
    printf("b = %d\n" , b);
    return 0;
}

```