23 feburary, 2023 Thursday

PROBLEM SOLVING & PROGRAM PLANNING:

⇒ Introduction:

you will not be able to solve a problem if you don't known the steps involved in solving that problem.
The same principle applies to writing a program. A programmer can't write instruction untill and unless, he don't know how to solve the same manually.

⇒ STEPS:

There are Steps Involved in solving a problem. A computer can't the problems on its own. One has to provide Steps → Solve that problem.

Programmer → write the Solution → Computer → follow
(In Simple operation)

* Understanding the Problem. (what Problem is)
* Analysing the Problem. → (ways)
* Developing the Solution → (Diagrams & flow charts)
* Coding and Implementation (Any language).

Space & Time →
↳ (Less) ⟶ To run the program → Solve Problem

⇒ **understanding the Problem:** what the problem is
Project → objectives of the problem
(to make)

Means understanding what the Problem is about.

⇒ **Analysing the Problem:** ~~After~~ analysing the problem, means we have to find the solution of that problem which is given tous.

⇒ **Developing the Solutions:** After then, we have to develop the solution of that Problem either by using diagrams, algorithms or flow-chart

⇒ **Coding & Implementation:** After then, we can do coding in any language and gets the output we desired.

* **For Planning:- (Program)**
The various tools collectively referred to as program design tools, that helps in progam planning and that are -

- Algorithm
- Flowchart
- Pseudocode

⇛ **ALGORITHM:** A set of squential steps written in simple Language to solve a given problem.

Important factor → Time
Less Time - Algorithm → Accepted

- Identification of input
- Identification of output
- Identification of operations.
- Processing Definiteness
- Processing Finiteness
- Processing Effectiveness

⇒ Properties of Algorithm

- Definiteness
- Finiteness
- Effectiveness
- Generality
- Input/output

⇒ To Find the ~~sum~~ average of ~~two~~ three numbers c, a and b.

1. Read the numbers a, b, c
2. Compute the sum of a, b & c
3. Divide it by 3
4. Store the result in variable d
5. Print the value of d
6. Stop

⇒ Principle (P), Rate of Interest (R), Time (T)

⇒ Area of Δ.
⇒ ~~Area~~ of Simple Interest.

⇒ **Algorithm to find area of Δ.**

- Start
- Read the variables b, h
- Compute the area of Δ by using the formula-
  $$Area = 1/2 \times b \times h$$
- Store the result in the Area
- Print the value of Area
- Stop.

⇒ **Algorithm to find the Simple Interest**

- Start
- Read the variables P, R, T
- Compute simple Interest by using the formula-
  $$SI = \frac{P \times R \times T}{100}$$
- Store the result in SI
- Print the value of SI
- Stop.

⇒ Write an Algorithm to find the Largest of three no. X, Y & Z.

```
1. Read the number X, Y and Z
2. if (x>Y)
       Big = X
   else
       Big = Y
3. if (Big < Z)
       Big = Z
4. Print the largest number i.e. Big
5. Stop.
```

- **FLOWCHART:** Step-step diagrametic repre-sentation to solve a particular problem → Flowchart.

Advantages] Flowchart + Algorithm
Properties]

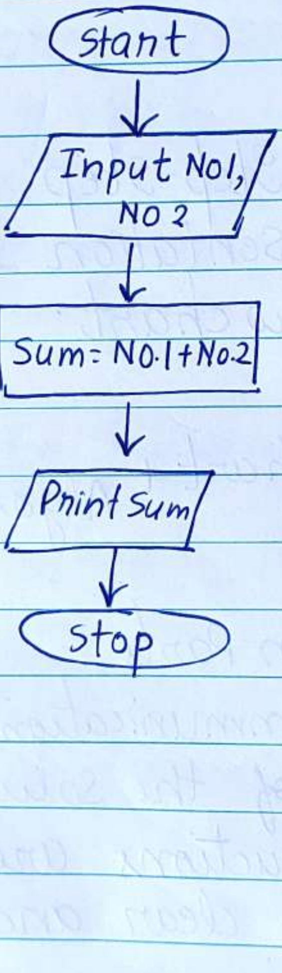
1. Break the problem in Parts
2. Good means of communication with user
3. Permanent Record of the solution.
4. The actual instructions are written within the Shapes using clear and concize state-ments.

The shapes are connected by direction lines to indicate the Squence in which instruction are to be executed.

⇒ Symbols for Flowchart

- Oval ⬭ : used for start & stop
- ▱ : Input & output
- ▭ : operations
- ◇ : Decision Making
- ⇄↓ : flow Lines

* **Flowchart**: To find Sum of the two numbers



27 feburary, 2023 Monday.

⇒ INTRODUCTION:

- C language ⟶ Dennis Ritchie $\xrightarrow{at}$ Bell Lab

UNIX OS ⟶ written in C language.

BCPL ⟶ later language → Martin Richards
(Basic complied Programming language)

ANSI
C Language → standardized in →1989.

⇒ ANATOMY:

Level of Interactions
- Low Level Language
- High Level Language

⇒ LOW LEVEL LANGUAGE:

⇒ Low LL → Machine language
↳ understand
(Binary & Assembly Language)
↳ understand by
Microprocessor

- Hardware dependent
- Not Portable
- Not an Easy job → knowledge of computer
- Architecture

⇒ HIGH LEVEL LANGUAGE:

FORTAN, BASIC, PASCAL, COBOL → Better programming efficiency.

⇒ English statements → Easy to understand
⇒ Portable
⇒ Not Hardware dependent

⇒ C : MIDDLE LEVEL LANGUAGE:

The C language lies b/w two categories. It has better environment than both the other level language. Middle Level language.

* Faster Access to all the things.
* The C language has all the elements as of any other modern high-level language.

⇒ CHARACTERISTICS OF C LANGUAGE:

1. Wide variety of Problems → C language.
2. No Rigid Format
3. Case Sensitive
4. Rich sets of operators (+, -, /, x, modolus)
5. Pointers → Memory location.
6. Portable

7. Own library function → Developed.

➜ Some features of C language are:

1. C program are very efficient and has fast Execu-tion Speed.
2. C language is rich in built-in functions or library functions
- C language programs are highly portable. Portability means - a C program written in one Environment Can be executed in another environment.
- C Language is structured Programming language. Structured programing means it has different modules and blocks.
- C Language is very Simple to learn and use.
- C Language is Middle level Language.
- C language has an important facility ca-lled extensibility. It means you can write your own file or functions & include in other programs.
- C language compiler gives reliable and accu-rate results. It has facility of warning, which guides for better & efficient progr-amming.
- C language is used to develop graphics soft-ware by using graphics programming.

⇛ APPLICATIONS:

1. UNIX operating system → C language
2. GUIs → Graphic Programming
3. Network Software to implement different Communication protocols etc.
4. Embedded Systems where C routines are interfaced high speed assembly Language routines and the resulting code is stored in ROM chip which is a part of the embedded System.

⇛ STRUCTURE OF C PROGRAM:

```
Section 1 : Comments
Section 2 : Preprocessor Directives
Section 3 : Global Declaration
Section 4 :
        Void Main ( )
          {
            Local Declaration; Statements:
          }
Section 5: Other functions are Required
```

EXAMPLES:

The Examples of the program is given below–

comments

```
// To print your name in C
#include <stdio.h>
void main ()
{
printf ("Hello");
}
```

⇒ Improve Readability of the program & it never helps to runs the program
(Comments)

- /* ______________
  ______________ */
  ↳ Multiline comment
- //______________
  ↳ Single line comment

⇒ PRE PROCESSOR DIRECTIVES:

- header files → used → Define ↓
  - Stdio.h → Extension for header File.
  - ↳ Standard input and output
- Input & output functions ↓
- All are defined in this → we can used in comp. Program.

⇒ Without it, program can't run without it.

#include directives

⇒ GLOBAL DECLARATION:

Global Declaration:
used to Read the variables.

⇒ VOID MAIN ( ):

Void main() → Execute the function
helps to run the Program

No main function → No Execution.
firstly Execute this, then Execute Input & Output functions & check whether we have declared that in header file or not.

⇒ To Print Your Name

```
// To Print your Name
#include <stdio.h>
void main ()
  {
  printf ("Gunsimran Kaur");
  }
```

↓ This will show on the screen → Printf is used → which is an output function.

⇒

⇒ C Program → Squence of characters.
written → Compliers → interpreted by it.
Characters

⇒ ass-Kee → value → understandably by computer
ASCII A = 67
ASCII a = 97

- Letters
- Characters
- Special characters
- White space characters

⇒ **KEYWORDS**: words already reserved in C. Such as void, printf & their meaning is also fixed. (predefined)

* Building Blocks of C
* Must be written in lower case.

⇒ **IDENTIFIERS:**

1. first Alphabet must be the letter & It can be an underscore.
2. It must consists of letter, digits & under-scrore only.
3. only 31 significant characters are there.
4. Cannot use a keywords
5. " " " white space.

first letter should be small.

- myFile
- roll_no
- _ehk
- date_of_birth
- file10
- area
- ABcd

⟵ **Valid**

- my-File
- pin. code
- 12file

] ⟶ **Invalid**

DATA TYPES:

Basic Data type

- Int → Simple whole No → (Integer) → 2
- float → Real No. → 4
- char → A Single char → 1

⇒ Integer → short int — Small whole No — 2
int — Medium " " — 2
long int — Large " " — 4

- -32768 to +32767 → Range → Short int or int
- -2147483648 to 2147483648 → Range → Long int
- unsigned int → 0-65535
- unsigned long int 0 - 4294967295

⇒ FLOAT DATA-TYPE:

float (single precision)	4 bytes	3.4×10^{-38} to $3.4 \times 10^{+38}$
Double (double precision)	8 bytes	1.7×10^{-308} to 1.7×10^{308}
long double	10 bytes	1.1×10^{-4932} to 1.1×10^{4932}

⇒ CHAR DATA-TYPE:

char	1 byte	-128 to 127
unsigned char	1 byte	0 to 255

⇒ OPERATORS:

• Arithmetic operators	+, -, *, /, %
• Relational operators	<, >, <=, >=, ==, =!
• Logical operators.	&&, ||
• Equality operators.	==, =!
• Assignment operators	=
• conditional operators	?:
• unary operators.	-, ++, --, size of etc
• Bitwise operator	%, |, ~, <<, >> etc

⇒ ARITHMETIC OPERATOR:

*	Multiplication	5*3 = 15
/	Division	10/2 = 5
%	Modulus (Remainder)	10%3 = 1

+	Addition	2+3 = 5
−	Subtraction	7−4 = 3

⇒ RELATIONAL OPERATOR:

<	Less than	a>b
>	Greater than	a<b
<=	Less than or Equal to	a<=b
>=	Greater than or Equal to	a>=b
==	Equal to	a==b
!=	Not Equal to	a!=b

⇒ LOGICAL OPERATOR:

!	Logical NOT
&&	Logical AND
||	Logical OR

• Logical And:

Expression 1	Expression 2	Exp.1 && Exp.2 (Interpretation)
False	False	False
False	True	False
True	False	False
True	True	True

• Logical OR:

Expression 1	Expression 2	Exp1 || Exp. 2
False	false	false
False	True	True
True	False	True
True	True	True .

- **Logical NOT:**

Expression	! Expression
True	False
False	True

⇒ ASSIGNMENT OPERATOR:

- Simple Assignment Statement
- Multiple Assignment statement
- Arithmetic Assignment Statement

⇒ Simple Assignment Statement

Variable = Expression;

a = b*c

⇒ Multiple Assignment Statement

V1 = V2 = V3 = V4 = Expression;

⇒ Arithmetic Assignment statement

variable Arithmetic operator = Expression;

⇒ COMMA OPERATOR:

C uses the comma operator in two ways. The first use of comma operator is to declare more than one variable in a single line. It acts as a seperator in the variable declaration.

int x, y, 3:

⇒ CONDITIONAL OPERATOR:

For conditional operator;

expr. 1 ? expr 2 : expr. 3

For Example: c = (a>b) ? a:b;

This means, if a>b then c becomes equal to a, otherwise c becomes equal to b.

⇒ UNARY OPERATOR:

C provides an increment operator (++) & a decrement operator (--) to increase & decrease the value of operand by 1.
These are unary operator & require only one operand to operate.

a) Prefix Notation: operator precedes the variable
++a, --- a

b) Suffix Notation: Operator follows the variable

Post- i++ ++i → Pre --i

a++ , a---

i++
↳ Increasing Value of i by 1

FOR EXAMPLE:

a=5 ;
b= ++a;

i--
↳ Decreasing value by 1

In this case, the b= 6 & a=6

a= 5 ;
b = a++

In this case, a=6 & b=5

Note;
In case of prefix Increment/Decrement operator (eg ++i or --i), the value of the variable will be first incremented or decremented before used in the expression.

In case of Suffix Increment/Decrement operator (eg i++, i--), the current value is used in the expression & after then value is Incremented or Decremented by 1.

⇒ write following Programs -
float pi = 3.14.

- Add two numbers
- Subraction, multiplication
- compute quotient & remainder %
- Area of circle (√)
- Area of Rectangle -
- Perimeter, SI

@ccgndec

1. To find the sum of two numbers

```
#include <stdio.h>
  void main ()
   {
   int a=5, b=3, c;
     c=a+b
  printf ("Sum is %d", c);
   }
```

But if we want to change the input again and again → Scanf,

But here the values are fixed → Static programming

2. To find the sum of two numbers using scanf function.

```
# include <stdio.h>
   void main ()
     {
    Int a,b,c
   printf (" Enter a and b");
   Scanf ("%d %d", &a, &b);
      c= a+b
  printf ("Sum is % d", c);
    }
```

Firstly we will provide Input, after then operations

- %d → int
- %f → float
- % c → Character

& - Ampersent
Address operator

⇒ when int a;
$\longrightarrow$ The space is given to a (Backside)

$\boxed{5}$
a ↘ 2 bytes

When we give the value to a, then that value will be filled in the box.
Seperate spaces will be given to a, b & c.

3. **Write a Program to subnact two numbens.**

```
# include <stdio.h>
    void main()
     {
     int a,b,c;
  Printf (" Enter a and b");
  Scanf (".|.d .|.d", &a, &b);
  Print    C = a-b
  Printf (" Subraction is .|.d", c);
       }
```

↳ Decision making statements
↳ loop control statements
↳ Jumping statements

⇒ DECISION MAKING STATEMENTS:

The decision making statements that alter the normal Squential Execution of the program depending on test condition.

- If
- If else
- If Nested if
- Else if ladder

⇒ operators used are basically Relational <, <=, >=, >, ==, =!

⇒ To find the largest among two numbers.

```
# include <stdioh>
     Void main()
          {
         int a,b;
Printf ("Enter a and b");
Scanf ("%d %d", &a , &b);
        if (a>b) ——→ condition.
          {
     prinf ("a is greater");
          }
             }
```

If a>b, only then It will print a is greater Otherwise program will be not executed.

2. To find largest among two numbers using if-else statement.

```
# include <stdio.h>
    void main ()
      {
      int a,b;
   printf ("Enter a and b");
   scanf ("%d %d", &a & b);
     if (a > b)
       {
    printf (" a is greater");
       }
     else
       {
    print f (" b is greater");
       }
        }
```

At a time, only one function will be executed either if will be or else will be.

3. To find largest of three numbers.

```
#include <stdio.h>
   Void main ()
   {
   int a,b,c;
printf (" Enter a,b and c");
Scanf ("%d %d %d", &a, &b, &c);
    if (a>b)
     {
      if (a>c)
       {
    printf(" a is largest");
       }
    else
      {
      printf ("c is largest");
      }
     }
  else
    {
     if (b>c);
    {
  printf ("b is largest");
    }
   else
  ~~p~~ {
   print f (" c is largest");
    }
     }
      }
```

1. To find the Multiplication of two numbers.

```
# include <stdio.h>
      void main ()
        {
        int a,b,c;
      Printf (" Enter the value of a and b");
      Scanf (" %d %d", &a, &b);
          C= a* b
      Printf (" Product is %d", c);
        }
```

2. To find the quoitent.

```
# include <stdio.h>
      void main ()
        {
        int a,b,c;
      Printf ("Enter the value of a and b);
      Scanf (" %d %d", &a, &b);
        C = a/b
      Printf (" quoitent is %d", c);
        }
```

3. To find the Remainder.

```
# inculde <stdio.h >
      void main ()
        {
        int p,q,r
Printf (" Enter p and q");
Scanf (" %d %d", &p , &q);
    r= p%q
Printf (" remainder is %d, r);
     }
```

4. To find the Area of the Rectangle.

```
#include <stdio.h>
  void main ()
    {
     int l, b, Area
  printf (" Enter the value of l and b");
  Scanf (" %d %d", &l , &b);
     Area = l*b
  Printf (" Area of Rectangle is %d", Area);
      }
```

5. To find the Perimeter of the Rectangle.

```
#include <stdio.h>
  void main ()
     {
      int l, b, perimeter
  printf (" Enter l and b");
  Scanf (" %d %d", &l , &b);
    Perimeter = 2(l+b)
  Printf (" Perimeter of Rectangle is %d", Perimeter);
      }
```

6. To find the area of circle.

```
#include <stdio.h>
    void main ()
      {
~~printf(~~ int r, area;
        float pi = 3.14;
  Printf ("Enter the value of r");
  Scanf ("%d", &r);
   area = pi*r*r;
  printf ("Area is %d", area);
      }
```

7. Find the greatest among three numbers using elseif ladder.

Else if ladder.

```
#include <stdio.h>
    void main()
      {
      int a,b,c;
  printf (" Enter the value of a, band c);
  scanf ("%d %d %d", &a, &b, &c);
  if (a>b && a>c);
   {
   printf ("a is largest");
   }
  else if (b>a && b>c);
   {
```

```
printf ("b is largest");
  }
else
{
printf (" c is largest");
  }
   }
```

Precedence
Assosiativity

a, b, c
7, 6, 8

⇒ a>b   a>c
7>6   7>8
↳ True   ↳ False ⟶ False

⇒ b>a   b>c
6>7   6>8
False   False ⟶ False

✓1. write a Program to check whether the number is even or odd.
✓2. Write a Program to check whether a number is positive or negative.
✓3. write a Program to check whether a year entered is leap year or not.
4. write a Program to print candidate is eligible for vote if the age is greater than or equal to 18 and less than ~~1~~80.
(Nested if)

5. write a program to print the grade of the Student according to the marks entered using else if ladder.

1. Write a program to check whether the number is even or odd.

```
#include <stdio.h>
      void main()
        {
      int a,b;
   printf ("Enter the value of a");
   Scanf ("%d", &a);
    b = a%2;
     if (b==0)
       {
     printf ("Number is even");
       }
     else
       {
     printf ("Number is odd");
       }
       }
```

2. Write a program to check whether a number is positive or Negative.

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

```
printf(" Enter the value of a");
Scanf ("%d", &a);
    if (a>0)
      {
    printf (" The number is positive");
      }
    else
      {
    printf (" The number is negative");
      }
    }
```

3. Write a program to check whether a year entered is leap year or not.

```
#include <stdio.h>
        void main()
          {
        int year,b
    printf (" Enter year");
    Scanf ("%d", year);
        b= year % 4;
    if (b==0)
        {
        printf ("The year entered is a leap year");
        }
        else
        {
    printf ("The year entered is not a leap year");
}
}
```

4. Write a program to print candidate is eligible for vote if the age is greater than or equal to 18 & ~~less~~ less than 80 using nested if.

```
#include <stdio.h>
void main()
{
int Age;
Printf ("Enter Age");
Scanf ("%d", Age);
if (Age >=18)
{
if (Age <80)
{
printf (" candidate is eligible");
}
else
{
printf (" candidate is not eligible");
}
}
else
{
printf (" not eligible");
}
}
```

5. Write a Program to print the grade of the Student according to the marks entered using else if Ladder.

```
# include <stdio.h>
     void main ()
      {
      intmarks;
   printf (" Enter Student's marks");
   Scanf ("%d", marks);
      if (marks >= 90 && marks <= 100)
         {
      printf (" The grade is A+");
         }
   else if (marks >= 80 && marks < 90)
      {
    printf (" The grade is A");
      }
   else if (marks>=70 && marks< 80)
      {
    print f (" The grade is B+");
      }
   else if (marks >= 60 && marks < 70)
      {
     printf (" The grade is B");
      }
   else if ( marks >= 50 && marks < 60)
         {
      print f ("The grade is C+");
         }
else if (marks >= 40 && marks < 50)
    {
  printf (" The grade is  c");
    }
else
   {
  printf ("Fail");
   }
  }
```

SWITCH STATEMENT:

```
# include <stdio.h>
      void main()
       {                        char c
       int a,b,c;               'a'  'B'
  Printf ("Enter a,b,c");       'A'  '+'
  Scanf ("%d %d %d", &a ,&b, &c);
  Switch(c) → choice
    {
  Case 1: → Not in single quote
  {
  printf ("Sum is %d", a+b);
   break; → for breaking the flow
  }
  Case 2: → Coln
  {
  printf (" Subraction is %d", a-b);
   break;
  }
  Case 3 :

  Case 4 :
```

For char
Case 'a'

[only
Single
Statement –
No Need of
{ }]

faster than
else if ladder

```
default:
printf ("Invalid choice");
  }
  }
```

Simple (calculator)

Switch statement is similar to the else if ladder. But else if ladder all the conditions are checked while in switch statement, we directly switch to desired condition.

⇒ LOOP CONTROL STATEMENT:

↳ while loop
↳ Do while loop
↳ For loop.

⇒ WHILE LOOP:

Print Natural No. from 1-10

```
#include <stdio.h>
      void main()
       {
      int i=1;
   while (i<= 10)
     {
   printf ("%d", i);
      i= i+1;
       }
```

Repitetive Type of work

$\xrightarrow{}$ Initial Value

- Intialization.
- Condition
- Increment | Decrement

  ↳ For Decrement (100 to 1)

⇒ FOR LOOP:

```
#include <stdio.h>
    void main ()
        {
        int i;
    for (i=1; i<=10; i++)
        {
      printf ("%d", i);
        }
        }
```

for Increment | Decrement:

• ⇒ int x = 5, y; → Pre Increment

y = ++x;
↳ Firstly Add & Assign

Pre Increment

y = x++;
↳ Firstly Assign & then Add.

Post Increment

• ⇒ =, == → (Equality operator)

= (Assignment operator)

== ↳ (For checking the LHS & RHS equality)

⇒ DO-WHILE LOOP: (Print 10 natural Numbers)

```
#include <stdio.h>
 void main ()
   {
   int i=1;   →(Counter)
   do
   {
  printf("%d", i);
   i++;
   }
  while (i<=10);
  }
```

→ (Important).

Q Difference between while and do while loop.

Ans: In Do-while loop, there is not condition Checking for i=1. In this case one time, loop will execute. So, this is an exit control loop while the while loop is entry control loop, in this condition will be checked first.

⇒ Like if, loops can also be nested;
for loop is the most convient, popular & versatile loop.

⇒ NESTED LOOP STATEMENTS:

⇒ Printing of the Matrix

$$\begin{array}{ccccc} & (1,1) & (1,2) & (1,3) & \\ (2,1) & * & * & * & (2,3) \\ & * & (2,2)\,* & * & \\ (3,1) & * & * & * & (3,3) \\ & (3,2) & & & \end{array}$$

⟶ Pattern of stars.

⇒

```
# include <stdio.h>
        void main ()
        {
        int m, n, i, j;
      printf (" Enter rows and columns");
      Scanf ("%d %d", &m, &n);
      for (i=1; i<=m, i++)
        {
      for(j=1, j<=n ; j++)
        {
(First → this   printf ("*");
loop will be    }
falsed)      printf ("|n");
             }
            }
```

$\underline{m}$ → No of Rows

$\underline{n}$ → No of columns

for (i=1; i<=m, i++) ⟶ For Rows

for(j=1, j<=n ; j++) ⟶ for columns

printf ("|n"); ⟶ newline

After then the value of i, will be Increased.

⇒ Printing the Matrix.

$$\begin{bmatrix} * & & & \\ * & * & & \\ * & * & * & \\ * & * & * & * \end{bmatrix}$$

Right-angled Δ.

```
for (i=1; i<=m; i++);
{
    for (j=1; j <=1, j++);
    {
    printf ("*");
    }
    printf ("|n");
    }
}
```

1. write a program, to print the factorial of a Number.
2. write a program, to print the Multiplication table of a Number.
3. write a program, to print the cube of a number upto n natural number.
4. write a program, to print the reverse of a number.

   (123 → 321)
5. write a Program, to check whether a number is palindrome or not.

   121

   $\textcircled{2}3\textcircled{2}$ $\xrightarrow{}$ Same → After reading from forward or Backward
6. Write a Program, to check whether the number is armstrong or not.

   153 $\quad 1^3 + 5^3 + 3^3$

   $1+125+27$
   $= 153$

   ⟶ No. → digits → sum (cube)
   ⟶ same

1. Write a program, to print factorial of a number.

```
#include <stdio.h>
    void main()
      {
      int i, n, fac=1;
    ~~while~~ printf(" Enter  number");
         Scanf("%d", &n);
      for (i=1; i<=n; i++)
        {
         fac = fac* i;
        }
    printf ("%d",fact);
      }
```

2. write a program to print cube of a number upto n natural number

```
# include <stdio.h>
    void main()
     {
    int i, n, cube;
   printf ("Enter n");
   Scanf ("%d", &n);
  printf ("\nNumber  Square");
   for (i=1; i<=n; i++)
     {
    cube = i*i*i ;
printf("%d %d", i, cube);
}
}
```

3. Write a Program, to print reverse of a Number.

```
# include <stdio.h>
        void main ()
int r=0; rev=0, num, t;
printf ("Enter number");
scanf ("%d", &num);
    t = num;
    do
    {
      r = t%10
      t = t/10
      rev = (rev*10)+r;
    }
    while(t>0);
printf ("Reversed number is %d", rev);
```

4. Write a Program to check, whether a number is Palindrome or not.

```
# include <stdio.h>
       void main ()
int r=0, rev=0, num, t;
printf ("Enter number");
```

```
scanf ("%d", &num);
          t = num;
     do
     {
     r = t % 10
      t = t / 10
     rev = (rev * 10) + r;
      }
     while (t>0);
printf("Reversed number is %d", rev);
     if (num == rev)
  printf(" Number is Palindrome");
  else
   printf (" Number is not Palindrome");
      }
```

5. **Write a Program to print Multiplica-tion table of a number.**

```
     # include <stdio.h>
          void main ()
           {
        int i, n, table;
  printf (" Enter the value");
  Scanf ("%d", &n);
  for (i= 1, i<= 10, i++)
      {
    table = n*i
  printf ("%d %d %d", n, i table );
```