Functions_day

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

// Function to add two numbers

```
int add(int a, int b) {
    return a + b;
}

int main() {
    int num1, num2, sum;
    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);

    sum = add(num1,num2 ); // Calling the function

    printf("The sum is: %d\n", sum);
    return 0;
}
```

—---

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

// Function to add two numbers

```
int add()
{
    return 2 + 3;
}


int main() {
    int result = add(); // Calling the function
    printf("The sum is: %d\n", result);
    return 0;
}
```

—---

```
#include <stdio.h>
// Function to print a number

void printNumber(int n)
{
    printf("The number is: %d\n", n);

}

int main() {
    printNumber(5); // Calling the function
    return 0;
}
```

—---

```
#include <stdio.h>


// Function definition

int max(int a, int b)
{
    if (a > b) {
        return a;
    } else {
        return b;
    }
}

int main() {
    int num1, num2;
        printf("Enter two numbers: ");
    scanf("%d%d", &num1, &num2);
    // Function call to find maximum
    printf("The maximum number is: %d\n", max(num1, num2));
    return 0;
}
```

```
#include <stdio.h>


// Function definition

int square(int num)
{
    return num * num;
}

 int main() {

    int num, result;

     // Input from the user

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


    // Function call to calculate square
    result = square(num);


    // Output the result
    printf("The square of %d is: %d\n", num, result);
    return 0;
}
```

———————————————————————————————————————

```c
#include <stdio.h>


// Function to print "Goodbye!"

void sayGoodbye() {
    printf("Goodbye!\n");
}



int main() {
    sayGoodbye(); // Calling the function
    return 0;
}
```

How many times the Hi will be printed?

```c
#include <stdio.h>
int i;
int fun();



int main()
{
   i= 1;

    while(i)
    {
        fun();
        main();
    }
    printf("Hello\n");
    return 0;
}



int fun()
{
    printf("Hi");
}
```

```
#include <stdio.h>

float func(float age[]);


int main()
{
    float result, age[] = {23.4, 55, 22.6, 3, 40.5, 18};
    result = func(age);
    printf("%f", result);
    return 0;
}

float func(float age[])
{
    int i;
    float result, sum = 0.0;
    for (i = 0; i < 6; ++i)
    {
        sum += age[i];
    }
    result = (sum / 6);
    return result;
}
```

—--

```
#include <stdio.h>
int main()
{
    int i;
    for (i = 0; i < 5; i++)
    {
        int i = 10;
        printf("%d ", i);
        i++;
    }
    return 0;
}
```

—--

```
#include <stdio.h>

void sort(int[],int);

int main() {

    int i,n,a[20];

    printf("enter the size of the array\n");

    scanf("%d",&n);

    printf("enter the elements of the array\n");

    for(i=0;i<n;i++)

        scanf("%d",&a[i]);

    sort(a,n);

    printf("sorted array\n");

    for(i=0;i<n;i++)

        printf("%d  ",a[i]);

    return 0;

}


void sort(int a[],int n )

{

    int i,j,temp;

    for(i=0;i<n-1;i++)

        for(j=i+1;j<n;j++)

            if(a[i]>a[j])

            {

                temp=a[i];

                a[i]=a[j];

                a[j]=temp;
            }
    return;
}
```

—--

Linear Search

```
#include <stdio.h>
void linearsearch(int[],int,int);
int main() {
    int i,n,a[20],ele;
    printf("enter the size of the array\n");
    scanf("%d",&n);
    printf("enter the elements of the array\n");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    printf("enter the element to search\n");
    scanf("%d",&ele);

    return 0;
}

void linearsearch(int a[],int n, int ele )
{
    int i, found=0;
    for(i=0;i<n;i++)
        if(a[i]==ele)
        {
            printf("%d is found at %d position\n",ele,i+1);
            found=1;
        }
    if(found==0)
        printf("%d not found\n",ele);
    return;
}
```

—---

strlen()

```
#include <stdio.h>
#include <string.h>
int main() {
    char str[20];
    int l;
    printf("enter a string\n");
    scanf("%s",str);
    l=strlen(str);
    printf("The length of %s=%d\n",str,l);
    return 0;
}
```

—---

strcpy()

```
#include <stdio.h>

#include<string.h>

int main()

{

     char str1[10],str2[10];

     printf("Enter source string\n ");

     scanf("%s",str1);

     strcpy(str2,str1);

     printf("Source String=%s\n",str1);

     printf("Destination String=%s\n",str2);

     return 0;

}
```

—---

Strcat()

```
#include<string.h>

int main()

{

     char str1[10],str2[10];

     printf("Enter two strings\n ");

     scanf("%s %s",str1,str2);

     strcat(str1,str2);

     printf("Concatenated String=%s\n",str1);

     return 0;

}
```

STRCMP()

```
#include <stdio.h>

#include<string.h>

int main()

{

     char str1[10],str2[10];


     printf("Enter two strings\n ");

     scanf("%s %s",str1,str2);

     //if(0==0)

if(strcmp(str1,str2) == 0)

 {

printf("Both the strings are identical\n");

printf(" Strcmp returns %d ",strcmp(str1,str2)
);


     }

     else

        printf("Both the strings are not
identical\n");

     return 0;

}
```

STRREV()

```
#include <stdio.h>

#include <string.h>


int main() {

    char str[20];

    printf("enter a string\n");

    scanf("%s",str);

    strrev(str);

    printf("Reverse =%s\n",str);

    return 0;

}
```

This program works only in TURBO C

CASE study -1

There are basically two types of functions:
Library functions Ex. printf( ), scanf( ) etc.
User-defined functions Ex.
fibonacci( ), sum( ) etc.

- Any function can be called from any other function
- A function can be called any number of times
- A function can call itself. Such a process is called 'recursion'.

```c
#include <stdio.h>

// Function declarations
void greetUser();
int add(int a, int b);
int multiply(int a, int b);
int findMax(int a, int b);

int main() {   // ----> CALLING FUNCTION
    int num1 = 10, num2 = 5;
    int sum, product, bigger;

    greetUser();  // Just a greeting

    // Call functions that return values
    sum     = add(num1, num2);   //---> CALLED function
    product = multiply(num1, num2);
    bigger  = findMax(num1, num2);

    // Display results
    printf(" You gave me: %d and %d\n", num1, num2);
    printf(" Sum: %d\n", sum);
    printf(" Product: %d\n", product);
    printf(" Bigger number: %d\n", bigger);

    return 0;
}

// Function definitions

void greetUser() {
    printf(" Hello! I'm your friendly calculator.\n");
    printf("Let's do some simple math with two numbers.\n\n");
}

int add(int a, int b)
{   int C;
   C = multiply(a,b);
    return C ;
}

int multiply(int a, int b) {
    return a * b;
}

int findMax(int a, int b) {
    if (a > b) {
        return a;
    } else {
        return b;
    }
}
```

Binary Search in Recursive form

```
 #include <stdio.h>

int binarySearch(int arr[], int low, int high, int key) {

    if (low <= high)

    {        int mid = (low + high) / 2;

        if ( arr[mid] == key )

                return mid;

        if ( arr[mid] > key )

             return binarySearch(arr, low, mid - 1, key);

        return binarySearch(arr, mid + 1, high, key);

    }

    return -1;

}

int main() {

    int arr[] = {2, 3, 4, 10, 40};

    int n = 5;

    int key = 10;

    int result = binarySearch(arr, 0, n - 1, key);

    printf("Element found at index: %d\n", result);

    return 0;
```

}

—---

Case 2 : USer Authentication.

—--

```
#include <stdio.h>
#include <string.h>

void displayWelcomeMessage();
int authenticateUser();
void performUserActions();
void displayLoginFailure();

int main() {
    displayWelcomeMessage();

    int login_success = authenticateUser();

    if (login_success) {
        printf("\nLogin successful! Welcome.\n");
        performUserActions();
    } else {
        displayLoginFailure();
    }

    printf("\nProgram Exiting. Goodbye!\n");
    return 0;
}
```

```
void displayWelcomeMessage() {
    printf("==========================================\n");
    printf("  Welcome to AVANTHI COLLEGE ....l!  \n");
    printf("==========================================\n");
}

int authenticateUser()
{
    char username[20];
    char password[20];
    const char correct_username[] = "admin";
    const char correct_password[] = "securepass";
    int attempts = 0;
    const int max_attempts = 3;

    while (attempts < max_attempts)
{
        printf("\n--- Login Required ---\n");
        printf("Enter username: ");
        scanf("%19s", username);
        printf("Enter password: ");
        scanf("%19s", password);
 if ( strcmp(username, correct_username) == 0&& strcmp(password, correct_password) == 0 ) {
            return 1;
        } else {
            printf("Invalid credentials. Please try again.\n");
            attempts++;
            printf("Attempts left: %d\n", max_attempts - attempts);
        }
    }
    return 0;
}

void performUserActions() {
    printf("\n--- User Dashboard ---\n");
    printf("1. View Profile\n");
    printf("2. Change Settings\n");
    printf("3. Access Secret Data\n");
    printf("----------------------\n");
    printf("Performing important actions...\n");
    printf("Data loaded successfully.\n");
}


void displayLoginFailure()
 {
    printf("\nMaximum login attempts exceeded. Access denied.\n");
    printf("Please contact support if you believe this is an error.\n");
}
```

=---

Pointers

Pointer arithmetic–

```
#include<stdio.h>

void main()

{

int arr[] = {10, 20, 30};

int *ptr = arr;

printf("Value at ptr: %d. %u \n", *ptr,ptr);

ptr++;

printf("Value at ptr after increment: %d. at %u \n", *ptr,
ptr);

ptr++;
printf("Value at ptr after increment: %d %u \n", *ptr,ptr);

}
```

——

Program for swapping of two numbers using call by value method

Call by value

```
#include <stdio.h>

void swap(int,int);

int main() {

   int a,b;

   printf("enter two numbers\n");

   scanf("%d %d",&a,&b);

   swap(a,b);

}

void swap(int x, int y) {

   int temp;

   temp = x;

   x = y;

   y = temp;

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

   return;

}

Output:

enter two numbers

10 20
```

```
After swapping: 20 10
```

In the above example the values of a and b are passed to x and y. what ever the changes made to the x and y does not reflect to a and b.

7.10.2 Call by Reference

In this method, the addresses of the variables are passed to the formal parameters in the called functions, therefore this method of parameter passing is called call by reference or call by address or call by pointer.

The actual parameters are the addresses of the variables and the formal parameters are the pointers. Because the addresses in the function call are pointed by the pointers in the called function, the changes made to the formal parameters in the called function reflect to the actual parameters in the calling function.

This is the default parameter passing mechanism for passing pointers to the functions

Program for swapping of two numbers using call by reference

```
#include <stdio.h>

void swap(int *,int *);

int main() {

   int a,b;

   printf("enter two numbers\n");

   scanf("%d %d",&a,&b);

   swap(&a,&b);

   printf("After swapping: %d %d \n",a,b);

}

void swap(int *x, int *y) {

   int temp;

   temp = *x;
```

```
    *x = *y;

    *y = temp;

    return;

}
```

----

printf("After swapping in swap function %d %d \n",x,y);

CAll by reference 👍

```
#include <stdio.h>
void swap(int *,int *);
int main() {
   int a,b;
   printf("enter two numbers\n");
   scanf("%d %d",&a,&b);

  printf(" Befor swapping in main function a=%d \n b=%d \n",a,b);

      swap(&a,&b);

     printf("After swapping in main function a=%d \n b=%d \n",a,b);
}
void swap(int *x, int *y) {
   int *temp;

   *temp = *x;
   *x = *y;
   *y = *temp;

     return;
}
```

----

https://forms.gle/cb1MWQJzCNP4c9KR8

Home work

```
#include<stdio.h>
void main()
{
  int i,j;
 for( i=0; i<3; i++);
  for ( j=0; j<3; j++);
       printf(" i= %d , j=%d ", i,j);
  }
```

What will happen when i for loop has no ; at the end?

____

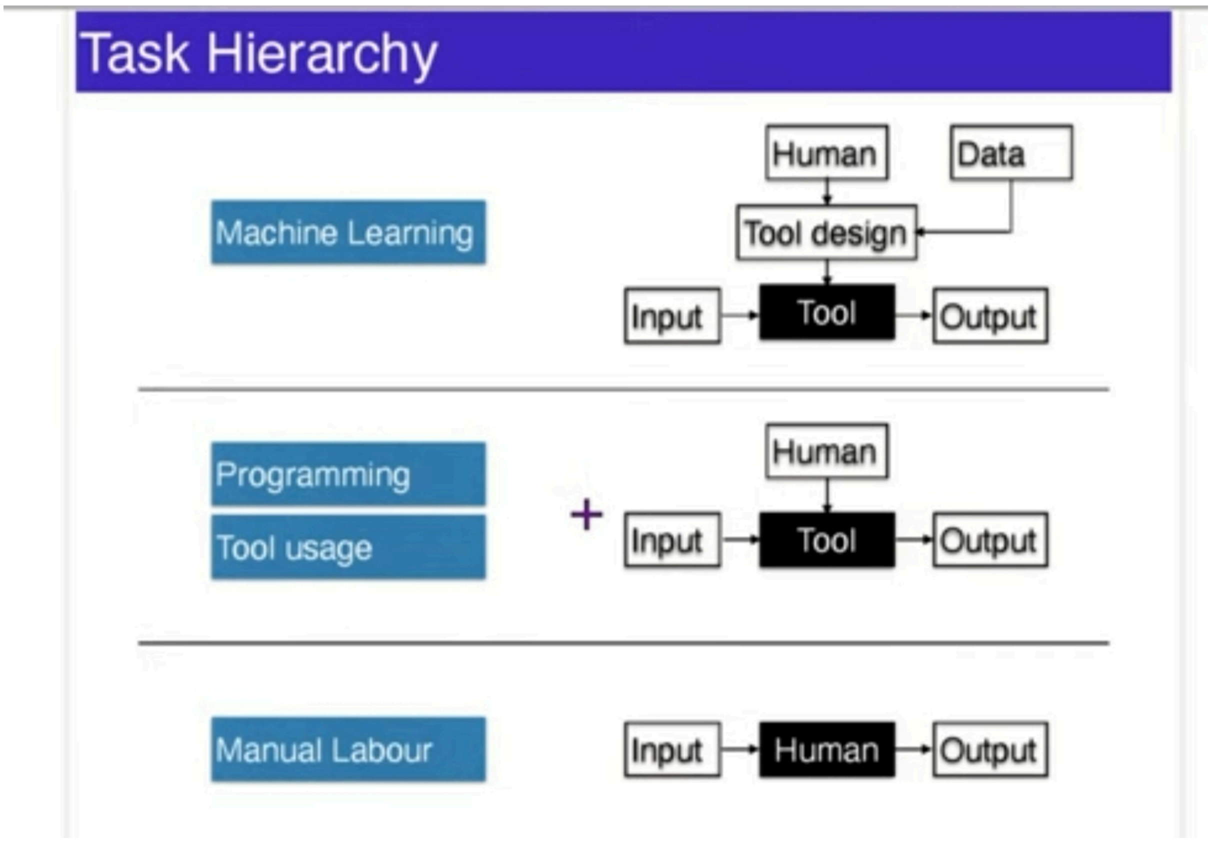
Home work -2

Who is this Technique or hierarchy Paradigm ?

-> ?

-> Programming

-> Humans



DMA

```
#include <stdio.h>

#include <stdlib.h>

int main() {

   int *a,sum=0,n,i;

   float mean;

   printf("enter the size of the arrayf\n");

   scanf("%d",&n);

   a= (int *) malloc(n*sizeof(int));

   printf("enter the array elements\n");

   for(i=0;i<n;i++)

   {

    scanf("%d",a+i);

    sum+=*(a+i);

   }

   mean=(float)sum/n;

   printf("Mean= %.2f \n",mean);

}
```

Ctrl+ shift + T

Checking the malloc positions

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   int *a,sum=0,n,i;
   float mean;
   printf("enter the size of the
array\n");
   scanf("%d",&n);

   a= (int *) malloc(n*
sizeof(int));

   for(i=0;i<n;i++)
   {
    printf(" %d ", *(a+i) );

   }
   printf("enter the array 
elements\n");
   for(i=0;i<n;i++)
   {
    scanf("%d",a+i);
    sum+=*(a+i);
   }
   mean=(float)sum/n;
   printf("Mean= %.2f \n",mean);
}
```

Calloc and also with memory locations

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   int *a,sum=0,n,i;
   float mean;

   printf("enter the size of the
array\n");

   scanf("%d",&n);


   a= (int *) calloc(n,
sizeof(int));


   for(i=0;i<n;i++)

   {

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



   }


   printf("\nenter the array
elements\n");

   for(i=0;i<n;i++)

   {
    scanf("%d",a+i);
    sum+=*(a+i);
   }
   mean=(float)sum/n;

// memory locations are
   for(i=0;i<n;i++)
   {
    printf(" %u  \n", a+i );

   }
   printf("Mean= %.2f \n",mean);
}
```

____

About the use of free()

____

```
#include <stdio.h>
#include <stdlib.h>
```

```
int main() {
   int *a,sum=0,n,i;
   float mean;
   printf("enter the size of the
array\n");
   scanf("%d",&n);

   a= (int *) calloc(n,
sizeof(int));

   for(i=0;i<n;i++)
   {
    printf(" %d ", *(a+i) );


   }

   printf("\nenter the array
elements\n");
```

```
   for(i=0;i<n;i++)
   {
    scanf("%d",a+i);
    sum+=*(a+i);
   }
   mean=(float)sum/n;
   // memory locations are
   for(i=0;i<n;i++)
   {
    printf(" %u  \n", a+i );


   }
   printf("Mean= %.2f \n",mean);


   free(a);
  // printf( "After free() %u
",a);
   printf( "After free() %d ",*a);
```

```
}
------------------------------------
```

Realloc program and printing the addresses also.

```
--------------------------------
#include<stdio.h>
//#include<process.h>
#include<malloc.h>

void main()
{
 int *p,i;
 p=(int *) malloc(3*sizeof(int));
if(p==NULL)
 {
  printf("nInsufficient memory");
 // exit(0);
```

```
  }

 printf("Enter three numbers:");
 for(i=0;i<3;++i)
  scanf("%d",p+i);

 for(i=0;i<3;++i)
  printf("%d ",*(p+i));

 //altering the memory
 p=realloc(p,5*sizeof(int));
if(p==NULL)
 {
  printf("nInsufficient memory");
  //exit(0);
 }
```

```
  printf("\nEnter two more
numbers:");
 scanf("%d%d",p+3,p+4);
 for(i=0;i<5;++i)
  printf("%d ",*(p+i));


  for(i=0;i<5;++i)
  printf("%u ",(p+i));


free(p);
}
```

Perfect Number

```
-> step1 : read n value
           Repeat 2,3,4
-> step2 : Iterate  From 1 to n/2
-> step3 : check if it is a factor
-> step4 : add it to sum variable


-> step4 : if sum==n
             Print "Perfect number"
          Else
           Print "NOT a Perfect number"
→ step 5: STOP
```

Count total number of digits are there in a number (N)

}

1234   ----   4

```
#include<stdio.h>
void main()
{
 int n,count=0;

printf(" Enter any number ");
scanf("%d",&n);

while( n!=0)
{
    n=n/10;
    count++;
}
printf(" The number of digits
are %d ", count);


}
```

—----------------------

—-----------------------

Write a C program to find the sum of all elements of each row of a matrix.
Example: For a matrix

```
4 5 6   4+5+6  →15
6 7 3   6+7+3  →16
1 2 3   1+2+3  →6
```

The output will be

```
15
16
 6
```

——------------------------------

```
1 2 3 4 5 6

1 2 3 4 5 6

1 2 3 4 5 6

1 2 3 4 5 6

1 2 3 4 5 6

1 2 3 4 5 6
```

```
#include<stdio.h>
#include<malloc.h>
void main()
{
 int n =0 ,i,k,count=0;
for(k=1;k<=6;k++)
 {
   for( i=1; i<=6; i++)
  {
     printf("%d ",i);
  }
   printf("\n");



 }


}
```

```
#include<stdio.h>

#include<malloc.h>

void main()
{
 int n =0 ,i,k,count=0;

 for(k=1;k<=6;k++)
 {
     for( i=1; i<=6; i++)
    {
        printf("*",i);
    }
     printf("\n");

}

}

output:
-------------------------
******
******
******
******
******
******
--------------------------
```

```
1 1 1 1 1
2 2 2 2 2
3 3 3 3 3
4 4 4 4 4
5 5 5 5 5
```

```
#include<stdio.h>

#include<malloc.h>

void main()
{
 int n =0 ,i,k,count=0;

 for(k=1;k<=6;k++)
 {
    for( i=1; i<=6; i++)
    {
       printf("%d ",k);
    }
    printf("\n");

}

}
```

```
0 1 1 1 1
2 0 2 2 2
3 3 0 3 3
4 4 4 0 4
5 5 5 5 0
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5

#include<stdio.h>

#include<malloc.h>

void main()
{
 int n =0 ,i,k,count=0;

 for(k=1;k<=6;k++)
 {
    for( i=1; i<=k; i++)
   {
       printf("%d ",k);


   }
    printf("\n");

}

}
```

```
#include <stdio.h>
int main() {
    int rows = 4;

 for (int i = 1; i <= rows; i++)
{

        for (int j = 1; j <= rows - i; j++)
        {
            printf(" ");
        }

        for (int k = 1; k <= (2 * i - 1); k++) {
            printf("*");
        }


        printf("\n");
    }
    return 0;
}
```

Output:

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

Write a C program to find the sum of all elements of each row of a matrix.
Example: For a matrix

```
                    4 5 6
                    6 7 3
                    1 2 3    The output will be
15
16
 6
```

```
→ Read elements into array (2D)

#include <stdio.h>
int main()
{ int i,j,a[3][3],sum=0;

printf("Enter elements into matrix\n");

for( i=0;i<3;i++)
  for (j=0;j<3;j++)
     scanf("%d",&a[i][j]);

for( i=0;i<3;i++)
{
    sum=0;
    for (j=0;j<3;j++)
    {
      sum=sum+a[i][j];
    }
    printf("%d \n ",sum*2);

}
    return 0;
}
```

Output

```
Enter elements into matrix
4 5 6
6 7 3
1 2 3
30
```

};

```
 32
 12
```

—--

Test Link - practice test 1

https://forms.cloud.microsoft/r/aBNmMXKpCc

—--

struct book

{

  int accessionno;

  char name[100];   —--------------> memory will not allocate

  char author[50];

  int pages;

  float price;

};

—---

struct book

{

int accessionno;

char name[100];   ——---> allocates memory

char author[50];

int pages;

float price;

} s;

```
—-----------------------------------

Int a;         —-> memory will be allocated
—-------------------
```

Write a program to display the division of a student by defining the structure student.

```
# include <stdio.h>

struct student

{

  int rollno;

    char name[25];

    int m1;

    int m2;
    int m3;
}s1;
int main( )
{
    float per;
    printf("Enter Rollno\n");
    scanf("%d",&s1.rollno);
    printf("Enter name of the student\n");
    scanf("%s",s1.name);
    printf("Enter marks in 3 subjects\n");
    scanf("%d %d %d",&s1.m1, &s1.m2, &s1.m3);
    per=(float) ( s1.m1 + s1.m2 + s1.m3)/3;
    printf("Student Details:\n");
    printf(" Roll No: %d\n Name: %s\n Percentage:
%f\n",s1.rollno,s1.name,per);
    return 0;
}
```

Output:

Example for Array of structures

```
//write a program to sort the list of students based on the
marks
# include <stdio.h>
# include <string.h>
```

```
struct student
{
    int rollno;
    char name[25];
    float per;
}s[20], temp;
int main( )
{
    int i,j,n;
    printf("enter number of students\n");
    scanf("%d", &n);
    for(i=0;i<n;i++)
    {
        printf("Enter rollno, name and percentage\n");
        scanf("%d %s %f",&s[i].rollno, s[i].name, &s[i].per);
    }

    for(i=0; i<n; i++)
       for(j=i+1; j< n-i-1; j++)
            if(s[i].per < s[j].per)
            {
                temp=s[i];
                s[i]=s[j];
                s[j]=temp;
            }
    printf("sorted list of students\n");

    for( i=0;i<n;i++)

        printf("%d  %s      %f\n",s[i].rollno, s[i].name,
s[i].per);


    return 0;

}
```

Output:

```
enter number of students

3

Enter rollno, name and percentage

1

anwar

97

Enter rollno, name and percentage

2

bhanu

67

Enter rollno, name and percentage

3


charan

55

sorted list of students

1  anwar      97.000000
```

```
2   bhanu       67.000000

3   charan      55.000000
```

FILES

//C programming code to open a file and to print it contents on screen.

```
# include <stdio.h>
 void main()
{
   char ch, file_name[25];
   FILE *fp;
    printf("Enter the name of file you wish to see\n");
   gets(file_name);

    fp = fopen(file_name,"w"); //write mode
    if( fp == NULL )
   {
      printf("Error while opening the file.\n");
      exit(0);
   }

    printf("Enter the contents of %s file :\n", file_name);
   while( ( ch = getchar() ) != '\n' )
      putc(ch,fp);
   fclose(fp);
   fp = fopen(file_name,"r"); // read mode
    if( fp == NULL )
      {
      print("Error while opening the file.\n");
      exit(0);
      }
    printf("The contents of %s file are :\n", file_name);
    while( ( ch = fgetc(fp) ) != EOF )
      printf("%c",ch);
    fclose(fp);

 }
```

Programs on Patterns

Programs on Patterns

// print Right angled triangle

```
#include <stdio.h>

int main() {
    int rows = 5;
    for (int i = 1; i <= rows; i++) {
        for (int j = 1; j <= i; j++) {
            printf("*");
        }
        printf("\n");
    }
    return 0;
}
```

Output:

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

—---

// printing a Square

```
#include <stdio.h>

int main() {
    int rows = 4;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < rows; j++) {
            printf("*");
        }
        printf("\n");
    }
    return 0;
}
```

Output:

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

—---

—---

Basic level

```
#include <stdio.h>

int main() {
    int n=5;

      for (int i = 1; i <= n; i++) {
        printf("%d ", i);
    }

    return 0;
}
```

Output:

```
1 2 3 4 5
```

—---

```
#include <stdio.h>

int main() {
    int n=5, sum = 0;
    for (int i = 1; i <= n; i++) {
        sum += i;
    }
   printf("Sum of first %d natural numbers is: %d\n", n, sum);

    return 0;
}
```

Output:

```
Sum of first 5 natural numbers is: 15
```

—---

```
// printing Even Numbers
#include <stdio.h>
int main() {
    int n=10;

    printf("Even numbers from 1 to %d are: ", n);
```

```c
    for (int i = 2; i <= n; i += 2) {
        printf("%d ", i);
    }

    return 0;
}
```

Output:

```
Even numbers from 1 to 10 are : 2 4 6 8 10
```

// printing Odd numbers

```c
#include <stdio.h>
int main() {
    int n = 10 ;
printf("Odd numbers from 1 to %d are: ", n);
    for (int i = 1; i <= n; i += 2) {
        printf("%d ", i);
    }
return 0;
}
```

Output:

```
Odd numbers from 1 to 10 are: 1 3 5 7 9
```

// Square of a number

```c
#include <stdio.h>

int main() {
    int n = 5 ;
    printf("Squares of numbers from 1 to %d are:\n", n);
    for (int i = 1; i <= n; i++) {
        printf("%d^2 = %d\n", i, i * i);
    }

    return 0;
}
```

Output:

```
Squares of numbers from 1 to 5 are:
1^2 = 1
2^2 = 4
3^2 = 9
4^2 = 16
5^2 = 25
```

```
// print sum of even numbers
#include <stdio.h>

int main() {
    int n, sum = 0;

    printf("Enter a number: ");
    scanf("%d", &n);

    for (int i = 2; i <= n; i += 2) {
        sum += i;
    }

    printf("Sum of even numbers up to %d is: %d\n", n, sum);

    return 0;
}
```
—--

```
// Numbers divisible by 3
#include <stdio.h>
int main() {
    int n;
    printf("Enter a number: ");
    scanf("%d", &n);
    printf("Numbers divisible by 3 up to %d are:\n", n);



for (int i = 1; i <= n; i++) {
        if (i % 3 == 0) {
           printf("%d ", i);
        }
    }
    return 0;
}
```
—--

// 5 Multiples

```
#include <stdio.h>

int main() {
    int n;

    printf("Enter a number: ");
    scanf("%d", &n);

    printf("Multiples of 5 up to %d are:\n", n);
    for (int i = 5; i <= n; i += 5) {
        printf("%d ", i);
    }

    return 0;
}
```

//**Counting digits in a number**

```
#include <stdio.h>
int main() {
    int n, count = 0;
    printf("Enter a number: ");
    scanf("%d", &n);
    for (int temp = n; temp != 0; temp /= 10) {
        count++;
    }
    printf("The number of digits in %d is: %d\n", n, count);

    return 0;
}
```

```
******  →6
* * * * * * →6
```

```
______________________
1 2 3 4 5 6
1 2 3 4 5 6
1 2 3 4 5 6
1 2 3 4 5 6
1 2 3 4 5 6
1 2 3 4 5 6
```

```
*
*
*
*


******
******
******
******


1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5


1 1 1 1 1
2 2 2 2 2
3 3 3 3 3
4 4 4 4 4
5 5 5 5 5

0 1 1 1 1
2 0 2 2 2
3 3 0 3 3
4 4 4 0 4
5 5 5 5 0

1 1 1 1 1
2 1 2 2 2
3 3 1 3 3
4 4 4 1 4
5 5 5 5 1
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5


1
2 3
```

```
4 5 6
7 8 9 10

1
2 3
4 5 6
7 8 9 10
```

// Print Pyramid

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

 for (int i = 1; i <= rows; i++)
{
        for (int j = 1; j <= rows - i; j++)
        {
            printf(" ");
        }
        for (int k = 1; k <= (2 * i - 1); k++) {
            printf("*");
        }
        printf("\n");
    }
    return 0;
}
```

Output:

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

// Print Numbers in a triangle

```
#include <stdio.h>

int main() {
    int rows = 5;
    for (int i = 1; i <= rows; i++) {
        for (int j = 1; j <= i; j++) {
            printf("%d ", j);
        }
        printf("\n");
    }
    return 0;
}
```

Output:

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

——--

——--

```
int main() {
   int i=1;
   for( printf( " Namasthe Boss \n ") ;  i< 3 ;  printf("   C chaltha  bhai %d \n",i++)  )
{
          if( i==1)
                 printf(" kya chaltha hai?\n " );
         else
               {
                  printf(" C  acha hai ?\n ");
                 break;
               }
}
printf(" acha acha.... " );
   return 0;
}
```

——--

——--

——--

——--

Strings_Home_work

—---—
---—--

// Printing the string

```
void main( )
{
char seminar[ ] = "birds_robbery" ;
int i = 0 ;
    while ( i < 13 )
    {
          printf ( "%c", seminar[i] ) ;
          i++ ;
   }
}
```

Output:

```
birds_robbery
```

—---

// Printing the string using %s format specifier

```
//The %s used in printf( ) is a format specification for printing out a string
void main( )
{
 char name[ ] = "Code_slashing" ;
 printf ( "%s", name ) ;
}
```

Output:

```
Code_slashing
```

—---

```
int main() {
   char str1[] = "Quiz-0-Assignment";
   char str2[] = {'Q, 'u', ‘i', 'z', '-', '0', '-', 'A', 's', 's', 'i', 'g', 'n', 'm', 'e', 'n', 't'};
   int n1 = sizeof(str1) / sizeof(str1[0]);
   int n2 = sizeof(str2) / sizeof(str2[0]);
   printf("n1 = %d, n2 = %d", n1, n2);
   return 0;
}
```

Output:

```
n1= 18  n2=17
```

—--

// Concatenating and Copying one string into another string

```
#include <stdio.h>
#include <string.h>

int main() {
   char str1[20] = "Hello", str2[20] = " world";
   printf("%s", strcpy(str2, strcat(str1, str2)));
   return 0;
}
```

Output:

```
Hello world
```

—--

// **Fill the blank to get the output as “ Hwi orea “**

```
int main() {
   int i;
   char s[] = "How is your exam";
   for (i = 0; s[i] != '\0'; ++i) {
      if ( —---------------------)
     {
         printf("%c", s[i]);
      }
   }
   return 0;
}
```

<u>solution :</u>

i%2==0

—--

```
#include <stdio.h>
#include <string.h>
```

```
int main() {
    static char str1[] = "_dd";
    static char str2[20];
    static char str3[] = "MIC";
    int i;
    i = strcmp(strcat(str3, strcpy(str2, str1)), "MIC_dd");
    printf("i = %d\n", i);
    return 0;
}
```

Output:

```
0
```

—--

//What could be the output of the following program?

```
#include <stdio.h>
#include <string.h>

int main()
{
    char p[] = "assignment";
    char t;
    int i, j;
    for (i = 0, j = strlen(p); i < j; i++)
    {
        t = p[i];
        p[i] = p[j - i];
        p[j - i] = t;
    }
    printf("%s", p);
    return 0;
}
```

Output:

Nothing will be printed

—--

// **Characters and functions**

```
void fu(int, int);
int main()
{
   char x = 67, y = 'C';
   fu(x, y);
   return 0;
}

void fu(int x, int y)
{
   printf("%d,%d", x, y);
}
```

Output:

```
 67  67
```

—---

// **Finding the length of the string without using the built in function**

```
int main() {
   char str[] = "Hello, world!";
   int length = 0;

   while (str[length] != '\0') {
      length++;
   }
   printf("Length of the string = %d\n", length);
   return 0;
}
```

Output:

```
Length of the string = 13
```

—---

```
void main( )
{
 char str1[ ] = { 'H', 'e', 'l', 'l', 'o' } ;
 char str2[ ] = "Hello" ;

 printf ( "\n%s", str1 ) ;
 printf ( "\n%s", str2 ) ;
}
```

<u>Output:</u>

```
Hello.
Hello
```

—---

—---

—---

—---