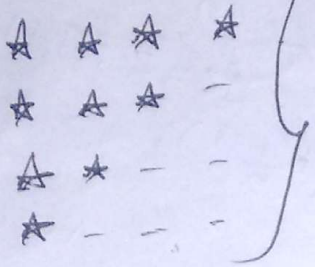
(I)

(a)

```
for(i=0; i<4; i++)
{ for(j=0; j<i; j++)
    pf("==");
  for(k=4; k>i; k++)
    pf("*-");
  pf("\n");
}
```

(b)



```
for(i=0; i<4; i++)
{
  for(j=4; j>i; j++)
    pf("*-");
}
```

(c)

```
for(i=0; i<4; i++)
{
for(k=0; k<i; k++)
    pf("  ");
for(j=4; j>i; j++)
  pf("*-");
}
```

(d)

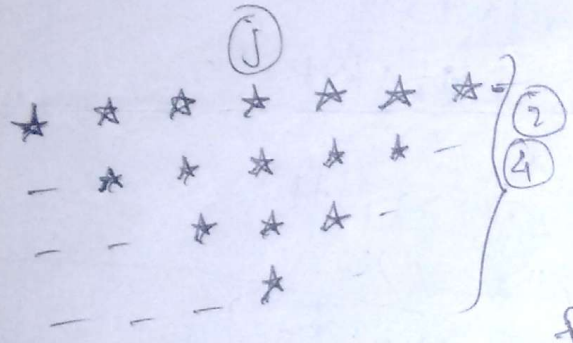
```
for(i=4; i>=0; i--)
  { for(k=i-1; k>0; k--)
      pf("  ");
    for(j=1; j<=i; j++)
      pf("*-");
  }
```

(e)

```
for(i=1; i<=5; i++)
{
 for(j=5; j>=1; j--)
   pf("* ");
 pf("\n");
}
```

n=5 i=1

```
for(i=1; i<=5; i++)
{
  for(k=n-1; k>=i; k--)
    pf(" ");
  for(j=1; j<=2; j++)
    pf("*");
}
```



```
j=7; k,l;
s=7/2;
i=j-s=7-3=4;
for(i=  ; i>=1; i--)
{
  for(k=5; k>=i; k--)
    pf(" ");
  for(l=1; l<=j; l++)
    pf("*");
  j=j-2;
  pf("\n");
}
```

① Prime No. :-

```
for (;num;)
{ k=num;
  for(i=2; i<num; i++)
  { if(num %.i==0)
      break;
  }
if(k==i)
  Pf(" %d ",num);
}
```

Start ①

Prime No.

② Armstrong No. :-

```
j=num;
for(;num;)
{
  s=num %.10;
  num=num/10;
  k=k+(s*s*s);
}
if(j==k)
  Pf("Armstrong No.");
else
  Pf("Not Armstng No.");
```

③ Perfect No. :-

```
int num, i=1, sum=0
Pf("Enter No.");
Sf("%d", &num);
while(i<num)
{ if(num%i==0)
  { sum=sum+i;
  }
  i++;
}
if(sum==num)
  Pf("Perfect No. ");
else
  Pf("Not Perfect No.");
```

④ Pelindrom No. :-

```
int n, r=0, rem, p;
while (n!=0)
{
    rem = n % 10;
    r = (r*10) + rem;
    n = n / 10;
}
if (n==r)
    Pf ("Pelindrom No.");
else
    Pf ("Not peli. No.");
```

(Reverse)

⑤ Reverse of No. :-

(Reverse the Digits of Number)

```
s=  ;
for ( ; num; s=(s*10)+(num%10), num=num/10);
```

⑥ Sum of Digits :-

(Add the Digits of Number)

```
s=0;
for ( ; num; s=s+num%10; num=num/10);
```

⑦ Check the Number is power of 2 or not :-

```
num & num-1 ? Pf("Not Power of 2") : Pf("Pow of 2");
```

⑧ Factorial of No. :-

```
fact = 1;
for ( ; num; )
{
    fact = fact * num;
    num--;
}
Pf("fact = %d", fact);
```

(9) check No. is ODD / Even :-

```
if(num%2==0)
  Pf("No. is EVEN.");
else
  Pf("No. is ODD ");
```

(10.) W. A. P. for n Febonaci Series :-

```
int a=0, b=1, num, c, p;
  Pf("num");
  Sf("%d", &n);
  Pf("%d %d ", a, b);
for(c=0; c<n; c++)
  {  p=a+b;
     a=b;
     b=p;
     Pf("%d ", p);
  }
```

(11.) W. A. P. for checking the given Year is leap Year or Not.

```
int i, year;
Pf("Ent. the yr");
Sf("%d", &yr);
if(yr % 400==0)
  Pf("leap Year");
else if(yr%100==0)
  Pf("Not leap Year.");
else if(yr%4==0)
  Pf("leap Year");
else
  Pf("Not leap Year");
return 0;
```

(12) W.A.P. for Implement a logic for Nibble Swaping.

```
unsigned int num, i;
Pf("Enter the Num: ");
Sf("%x", &num);
i = num >> 8 | num << 8;
Pf("num: %hx \n", i);
```

```
unsigned char ch = 0x3c;
Pf("Num: %d \n", ch);
ch = ch >> 4 | ch << 4;
Pf("num: %x \n", ch);
```

(13) Swap two No. :- `{ i ^= j ^= i ^= j; }`

Using temp. vari.

```
temp = i;
i = j;
j = temp;
```

Without using temp. vari.

```
i = i + j;
j = i - j;
i = i - j;
```

```
i = i * j;
j = i / j;
i = i / j;
```

using single line.

```
j = i + j - (i = j);
```

And (&&) → 11 → 1 } if any 0 ⇒ Ans → 0
OR (||) → 00 → 0 } if any 1 ⇒ Ans → 1
x'or (^) → 00 → 0, 11 → 0 } otherwise 1

(1) set a bit :- `num = num | 1 << pos;`

(2) clear a bit :- `num = num & ~(1 << pos);`

(3) Compliment a bit :- `num = num ^ 1 << pos;`

(4) check the Bit is set or clear :-

```
res = num >> pos & 1;
```

(5) Convert the Number into Binary format :-

```
for(pos = 31; pos >= 0; pos--)
    Pf("%d ", num >> pos & 1);
```

(6.) Reverse printing of Binary ~~Digit~~ Bits of Number.

```
for(pos=0; pos<=31; pos++)
    pf("%d", num>>pos&1);
```

(7.) Reverse the Bits of given No. :-

```
for(i=0, j=31; i<j; i++, j--)
{
    m = num >> i & 1;
    n = num >> j & 1;
    if (m != n)
    {
        num = num ^ 1 << i;
        num = num ^ 1 << j;
    }
}
```

(8.) Count How Many Bits are set in No. :-

```
for (pos=31; pos>=0; pos--)
    if (num >> pos & 1)
        c++;
pf("%d", c);
```

Count

(9.) Convert the Binary of float No. Using integer pointer :-

```
float f=23.5;
int *p, i;
ip = (int *)&f;
for(i=31; i>=0; i--)
    pf("%d", *ip >> i & 1);
pf("\n");
```

⑨ Find the LCM of two No. :-

```
int main()
{ int n1, n2, miniMutiple;
  Pf("Enter two +ve integer: ");
  Sf("%od %od", &n1, &n2);

  miniMultiple = (n1>n2)? n1 : n2;
  while (1)
  {
    if (miniMul.%n1==0 && miniMul.%n2==0)
    {
      Pf("the LCM of %od & %od is %od",
                    n1, n2, miniMul);
      break;
    }
    ++minimultiple;
  }
  return 0;
}
```

⑩ Convert the float No. into the Binary form using char pointer :-

```
int i, j;
float f=3.5;
char *cp;
cp=&f;
cp=cp+3;
for(i=0; i<4; i++)
{
 for(j=7; j>=0; j--)
   Pf("%d", *cp>>j&1);
 cp=cp-1;
}
 Pf("\n");
```

(*) Void my-strcpy (const char *, char *); (3)

```
main()
{
    my-strcpy(s,d);
    Pf("s=%os  d=%os\n", s,d);
}
Void my-strcpy(const char *s, char *d)
{
    int i;
    for(i=0; s[i]; i++)
        d[i]=s[i];
    d[i]='\0';
    return d[i];
}
```

(*) Void my-strncpy (const char *, char *, int);

```
main()
{
    my-strncpy(s,d,n);
    Pf("s=%os  d=%os\n", s,d);
}
void my-strncpy(const char *s, char *d, int n)
{
    int i;
    for(i=0; i<n; i++)
        d[i]=s[i];
    d[i]='\0';
}
```

(*) int my_strcmp (const char *, const
main()
{
int ret;
ret = my_strcmp(s1, s2);
if(ret == 0)
pf("equal");
else
pf("Not equal...\n");
}
int my_strcmp (const char *p, const char *q)
{ int i;
for(i=0; p[i]; i++)
{ if(p[i] != q[i])
break;
}
if(p[i] == q[i])
return 0;
else
return p[i] - q[i];
}
(*) int my_strncmp (const char *p, const char *q,
int n)
{ int i;
for(i=0; i<n-1; i++)
{ if(p[i] != q[i])
break;
}
if(p[i] == q[i])
return 0;
else return p[i] - q[i];
}

```
(*) void *my_strchr(const char *, char *);
main()
{
    p = my_strchr(s, ch);
    if (p==0)
    pf("char is not present\n");
    else
    pf("char is present\n");
}
void *my_strstr(const char *p, char *ch);
{
    int i;
    for(i=0; p[i]; i++)
    {
        if (p[i]==ch)
        return p+i;
    }
    return 0;
}
(*) int *my_strrchr(const char *p, char ch);
{
    int i;
    for(i=0; p[i]; i++);
    for( ; i>=0; i--)
    {
        if (p[i]==ch)
        return p+i;
    }
    return 0;
}
```

(*) void my_strcat (char *, const char *);

main()

{

my_strcat (s, d);

Pf("Source = %os Desti = %os \n", s, d);

}

void my_strcat (char *s, const char *d)

{ int i, j;

→ for (i=0; s[i]; i++);

for (j=0; d[j]; j++)

s[i++] = d[j];

s[i] = '\0';

}

(*) void my_strncat (char *s, const char *d, int n)

{ int i, j;

→ for (i=0; s[i]; i++);

→ for (j=0; j<n; j++)

s[i++] = d[j];

s[i] = '\0';

}

(4) ~~write a program~~

(4)

```
int *my_strstr (const char *, const char *);
main ()
{
    int ret;
    char s1[100], s2[100];
    Pf("Enter string s1:");
    Sf ("%s", s1);
    Pf("Enter string s2:");
    Sf ("%s", s2);
  ret=my_strstr(s1,s2);
 if(ret==0)
   Pf("(%d) string is not present..\n",
                                ret);
  else
   Pf("(%d) string is present..\n", ret);
}
int *my_strstr(const char *p, const char *q)
{ int i, j;
 for(i=0; p[i]; i++)
  { if (p[i]==q[0])
     { for (j=0; q[j]; i++,j++)
        { if (p[i]!=q[j])
            break;
        }
      if (q[j]=='\0')
        return p+i;
      i--;
  }
 return 0;
}
```

```
int my_strlen (const char *);
main()
{ char s[20];
  int ret;
  ret=my_strlen(s);
  pf("length=%d\n", ret);
}
int my_strlen (const char *p);
{ int i;
  for(i=0; p[i]; i++);
  return i;
}
```

Recursion

Factorial using Recursion:-

```
int fact (int);
main()
{
  ret=fact(num);
  pf("fact=%d\n", ret);
}
int fact(int n)
{ if(n)
    return n*fact(n-1);
  else
    return 1;
}
```

⋆ print string char by char

```
void print(char *);
main()
{ char s[] = "abcd";
   print(s);
   pf("\n");
}
void print(char *p)
{ if(*p)
  {
    pf("%c ", *p);
    print(p+1);
  }
}
```

⋆ print string in Reverse:-

```
void print_rev(char *p)
{ if(*p)
  { print(p+1);
    pf("%c ", *p);
  }
}
```

```
(*) int my_strlen(const char *);
main()
{ char s[20];
  int l;
  pf("Enter the strings:")
   sf("%s",s);
  l=my_strlen(s);
  pf("length=%d\n",l);
}
int my_strlen(const char *p)
{
  if (*p)
  return (1+my_strlen(p+1));
  else
  return 0;
}
(*) void my_strcpy(const char *p,
                   char *q)
{ if(*p)
  { *q=*p;
    my_strcpy(p+1,q+1);
  }
  else
    *q=*p;
{
```

(A) int *my_strchr(const char *, char);

main()

{ char s[20], ch;

int i;

pf("En the s ");

sf("%s", s);

pf("En the ch");

sf("%c", ch);

i = my_strchr(s, ch);

if(i==0)

pf("char is not present");

else

pf("char is present");

}

int *my_strchr(const char *p, char ch)

{

if(*p)

{ if(*p==ch)

return p;

else

return (my_strchr(p+1, ch));

}

else

return 0;

}

FILE Lect V ①

⑥

⊛ predefined copy cmd :-

(./a.out sf df)

```
#include <stdio.h>
main(int argc, char **argv)
{
    char ch;
    FILE *fp, *fd;
    if(argc!=3)
    {
        Pf("Usage: ./a.out sf df\n");
        return 0;
    }
    fp=fopen(argv[1],"r");
    if(fp==0)
    {
        Pf("Source file Not pre.\n");
        return 0;
    }
    fd = fopen(argv[2],"w");
    while((ch=fgetc(fp))!=-1)
    {
        fputc(ch,fd);
    }
}
```

⊛ predefined Copy cmd for Source to Multiple desti

(./a.out sf df1 df2 ....)

```
main(int argc, char **argv)
{ int i;
  char ch;
  FILE *fp, *fd;
  if(argc<3)
  { Pf("Usage: ./a.out sf df1 df2...\n");
    return 0;
  }
  fp=fopen(argv[1],"r");
  Pf(fp==0)
  { Pf("Source file is not present\n");
    return 0;
```

```
For(i=2; i<argc ; i++)
{
    fd=fopen(*argv[i],"w");
    while((ch=fgetc(fp))!=-1)
        fputc(ch,fd);
    rewind(fp);
}
}
```

⭐ Find out the size of file:-

```
main(int argc, char **argv)
{ int c=0;                    (./a.out fname)
  FILE *fp;
  if(argc!=2)
    { pf("Usage: ./a.out fname");
      return 0;
    }
  fp=fopen(argv[1],"r");
  if(fp==0)
    { pf("file Not pre.");
      return 0;
    }
  while(fgetc(fp)!=-1)
      c++;
  Pf("size=%d\n",c);
}
```

without Using Counter:- / without opening the File

① → fseek(fp, 0, SEEK_END);

② → l = ftell(fp);

(*) Search How Many time's a given char is present in a given file.

```
op> int main(int argc, char **argv)
    {
      FILE *fp;                          (./a.out fname ch)
      char ch;
      int  c=0;
      if (argc != 3)
      { pf("Usage: .la.out fname ch\n");
        return 0;
      }
      fp = fopen(argv[1], "r");
      if (fp==0)
      { pf("file is Not Pre.\n");
        return 0;
      }
      while ((ch=fgetc(fp)) != -1)
        if (ch == argv[2][0])
          c++;
      pf("count = %d\n", c);
    }
```

(*) Where the char is present Replace with another char. :-

```
op> int main (int argc, char **argv)
    {                                    (./a.out fname ch R_ch)
      File *fp;
      char ch, *p;
      int c=0, i;
      if (argc != 4)
      { pf("Usage: - - - .\n");
        return 0;
      }
      fp = fopen(argv[1], "r+");
      if (fp==0)
      { pf("file is Not Pr.\n");
      } return 0;
```

```
While (fgetc(fp) != -1)
    C++;
rewind(fp);
p=malloc(sizeof(ch));
```

(Size of file.)

(Allocat DMA)

```
i=0;
while((ch=fgetc(fp)) != -1)
    P[i++]=ch;
P[i]='\0';
rewind(fp);
```

→ (Copy the file into one array)

← otherwise both data can store previous & new (with Replace char)

```
For(i=0; P[i]; i++)
  if(P[i]==argv[2][0])
      P[i]=argv[3][0];
```

(Replacement in Array)

```
For(i=0; P[i]; i++)
    fputc(P[i], fp);
}
```

→ (copy the content of array into file).

⑧ Source file to Multiple Destination File

```
main(int argc, char **argv)
{ int i;
  char ch;
  FILE *fp, *fd;
  if(argc<3)
  { pf("Usage: ./a.out sf df1 df2 ...\n");
    return 0;
  }
  fp=fopen(argv[1], "r");
  if(fp==0)
  { pf("file is Not present\n");
    return 0;
  }
```

```
for (i=2; i<argc; i++)
{
  fd= fopen (argv[i], "w");
     while ((ch=fgetc (fp)) != EOF)
             fputc (ch, fd);
  rewind (fp);
  }
}
```

grep command :- #include <string.h>

```
main (int argc, char **argv)
{
  char s[50];
  FILE *fp;
     if (argc != 3)
      { pf ("usage: ./a.out string fname\n");
        return;
      }
     fp= fopen (argv[2], "r");
      if(fp==0)
       { pf (" file not present \n");
         return;
       }
  While (fgets (s, 50, fp))
   { if (strstr (s, argv[1]))
        pf ("%s", s);
   }
}
```

⭐ Replace Word with Reverse Word

```
main(int argc, char **argv)
{ int i,j,l;
  char rev[20], s[20], temp;
  FILE *fp;
  if(argc!=2)
  { printf("Usage:-la.out string fname\n");
    return;
  }
  fp=fopen("argv[1]","r+");
  if(fp==0)
  { printf("file not present\n");
    return 0;
  }
  strcpy(rev, argv[2]);        // Copy the string in one array
  l=strlen(rev);               // find length of string.
  for(i=0,j=l-1; i<j; i++,j--)
  { temp=rev[i];               // Reverse the string into the array rev[].
    rev[i]=rev[j];
    rev[j]=temp;
  }
  while(fscanf(fp,"%s",s)!=EOF)
    if(strcmp(s,argv[2])
    { fseek(fp,-l,SEEK_CUR);
      fprintf(fp,"%s",rev);
    }
}
```

(A) Merge two files :-

FILE Assignment

```
main(int argc, char **argv)
{
    FILE *sf, *sf1, *df;
    char p[100], p1[100];
    if (argc != 4)
    { Pf("Usage:-a.out sf sf1 df\n");
      return 0;
    }
    sf = fopen(argv[1], "r");
    sf1 = fopen(argv[2], "r");
    df = fopen(argv[3], "w+");
    if (sf==0 || sf1==0)
    { Pf("%s or %s is not present\n");
      return 0;
    }
    while (fgets(p, 100, sf) != NULL)
        fputs(p, df);
    fclose(sf);
    while (fgets(p1, 100, sf1) != NULL)
        fputs(p1, df);
    fclose(sf1);
    fclose(df);
    Pf("Done\n");
}
```

⭐ Reverse file contents :-

```
main( int argc, char **argv)
{
  if (argc != 2)
  {
    Pf("Useage: ./a.out fname \n");
    return 0;
  }
  char *p, ch;
  int c, i=0;
  FILE *fp;
  fp=fopen(*argv[1], "r+");
  if(fp==0)
  {
    Pf("file is not present");
    return 0;
  }
fseek(fp, 0, SEEK_END);
c=ftell(fp);
p=malloc(c+1);
rewind(fp);
while((ch=fgetc(fp))!=EOF)
  {
    p[i++]=ch;
  }
→ fclose(fp);
  fp= fopen(argv[1], "w");
  i=c;
  while(i>=0)
    fputc(p[i--], fp);
}
```

① Lower to upper & vise versa :- FILE Assignment 8

```
→ fp = fopen(argv[1], "r+");
  if (fp==0)
  {
    printf("file is not present");
    return 0;
  }
→ while ((ch=fgetc(fp)) != -1)
  {
    if (ch>='A' && ch<='z')
    {
      ch = ch ^ 32;
      fseek(fp, -1, SEEK_CUR);
      fputc(ch, fp);
    }
  }
```

only

② Capitalize first letter of Every Word :-

← Add <string.h>.

```
main(int argc, char **argv)
{
  int i,n;
  char ch[100];
  FILE *fp;
→ fp = fopen(argv[1], "r+");
  while(fscanf(fp, "%s", ch) != -1)
  {
    n = strlen(ch);
    if (ch[0]>='a' && ch[0]<='z')
      ch[0] = ch[0] ^ 32;
    for(i=1; i<n; i++)
    {
      if (ch[i]>='A' && ch[i]<='Z')
        ch[i] = ch[i] ^ 32;
    }
    fseek(fp, -n, SEEK_CUR);
    fprintf(fp, "%s", ch);
  }
```

→ Capitalize only second ~~word~~ letter of word:-

```
while (fscanf(fp, "%s", ch) != -1)
{
    n = strlen(ch);
    if (ch[0] >= 'A' && ch[0] <= 'Z')
        ch[0] = ch[0] ^ 32;
    if (ch[1] >= 'a' && ch[1] <= 'z')
        ch[1] = ch[1] ^ 32;
    for (i=2; i<n; i++)
    {
        if (ch[i] >= 'A' && ch[i] <= 'Z')
            ch[i] = ch[i] ^ 32;
    }
    fseek(fp, -n, SEEK_CUR);
    fprintf(fp, "%s", ch);
}
```

→ First letter Capital then alternate:-

```
while (fscanf(fp, "%s", ch) != -1)
{
    n = strlen(ch);
    if (ch[0] >= 'a' && ch[0] <= 'z')
        ch[0] = ch[0] ^ 32;
    for (i=1; i<n; i++)
    {
        if (i%2 == 0)
        {
            ch[i] = ch[i] ^ 32;
        }
    }
    fseek(fp, -n, SEEK_CUR);
    fprintf(fp, "%s", ch);
}
```

→ First letter Small then alternate :-

```
while (      )
{   n=strlen(ch);
→ if (ch[0]>='A' && ch[0]<='Z')
        ch[0]= ch[0]^32;
→ for(i=1; i<n;i++)
   {  if (i%2==0)
      {  ch[i]= ch[i];
      }
      else ch[i]=ch[i]^32;
   }fseek (fp, -n, SEEK_CUR);
    fprintf (fp, "%s", ch);
}
```

⋆ WC Command :-

```
main( int argc, char **argv)
{ if (argc!=2)
  { pf("Usage: .\amout fname");
    return;
  }
  FILE *fp;
  char ch, s[100];
→ int CHAR=0, LINE=0, WORD=0;
  fp=fopen (argv[1], "r");
  if(fp==0)
  { pf("file is not present");
    return 0;
  }
```

```
while(fscanf(fp,"%s",s)!=EOF)
        WORD++;
rewind(fp);
while((ch=fgetc(fp))!=EOF)
{   CHAR++;
    if(ch=='\n');
        LINE++;
}
Pf("LINE=%d WORD=%d CHAR=%d",
                LINE,WORD,CHAR);
}
```

9

* **Pointer:-** It is a one of the secondary data type which is used for storing the address of another data type, through which we can access the data indirectly.

* **Array:-** It is a one of the secondary data type which is a collection of similar elements which are in contigeous memory location.

* **Qualifiers:-** ① Volatile ② Const

→ It is used for Increase the Accuracy and Remove the Confusing.

* **Function:-**

→ Function is a set of instructions put together to perform a specific task.

* **Recursive Fun:-**

→ In one fun if we call the same fun, that fun is called Recursive fun.

Memory leak :-

→ If we lose the base address of dynamically allocated memory, that memory became's leaked

Solution :- i) Catch with two diffent pointer for two diffent Dynamic Memory.

ii) After using that Make it free. so other can use easily.

Dangling pointer :-

→ If a pointer pointing to Unreserved memory then this pointer is called Dangling pointer.

→ After freeying mem. if the pointer is still pointing to that freed memory then this pointer is called Dangling pointer.

Solution:- After using pointer, make that pointer as 'NULL' pointer.

Structure:- It is a one of the userdefined data type, which is a collection of different type's of Data or in contigeous Memory location.

Nested structure:-

→ If we creat a variable of one structure in another structure that is called Nested structure.

self referencial structure:-

→ If we creat a pointer as a member of same structure type that structure is called self referencial structure.

Structure Bit fields:-

→ It is a mechanism, where we can allocate a memory in the form of Bits.

Structure Peddings:-

→ Adding some extra byte then the required memory for a structure variable is called Structure pedding.

Avoiding structure paddings

i) Declare all char in one side, and short in one side

ii) Used # pragma pack (1)

★ Union :-

It is a special data type which allows to store different data types in the same memory location.

(In a structure all members are having an independent location where as the union members are sharing same mem. location)

Diff. of structure & Union :-

(1) The amount of memory required to store a structure variable is the sum of the size of all members. while in union, the amount of mem. required is always equal to that required by its largest member.

(2) In structure each member have their own memory space, but while in union one block is used by all the members of the union

★ typedef ():-

(1D) → It is one of the user defined datatype which is used to put another Name. to the existing type. So that Declarations becomes simple & easy to understand (alias name)

(→ The New name should Not be a keyword)

★ Macro :-

→ Macro is a small portion of the progeramme which is replaced with another portion by the preprocessor.

→ If we wants to defined a Macro used one of the directives (#define, #include, #pragma, #if, ...)

#define

→ # define Name body

→ info. passing to preprocessor.

→ Replacement is done.

→ Useful for anytype like:- data type constant etc.

typedef

→ #typedef oldname Newname

→ info. passing to translator

→ Replacement is not done. (use old as well as New Name)

→ only useful for data types.

Macro with Arg.

```
#define MUL(a,b) (a)*(b)
```

→ Macro Arg. Doesn't have any type.

→ Act's as a generic type

→ Writing macro put in only Capital latter.

→ If a task is smaller write a Macro

Fun with Arg.

```
int mul (int i, int j)
```

→ Fun Arg. having a specific data type.

→ Act as a specific data type

→ writing fun put it in small letter.

→ If task is bigger write a Fun.

malloc

→ prototype :-
Void *malloc(size);

→ Returns allocated memory's starting Add.

→ used for allocating Dynamic mem.

→ Ex:- int *p[5];
p=malloc(sizeof(int)*5);

calloc

→ prototype :-
void *calloc(nmemb, size);

→ same as malloc

→ same as malloc

→ Ex:- int *p[5];
p=malloc(5, sizeof(int));

⇨ After allocating memory that memory is not cleared or may be cleared.

→ After allocating mem. that memory set to Zero.

⇨ Malloc is faster compeare to Calloc because it will doing Allocating, cleared & Return.