DYNAMIC MEMORY ALLOCATION

Definition:

•Allocating a memory at run time is called dynamic memory allocation.

•C provides the following function for achieving it. Those are
•malloc(), calloc(), realloc().

•These functions are available in **stdlib** file.

•This memory allocation done in heap memory.

#include<stdlib.h>

malloc()

•This function is used for allocating the block of memory.

•If the memory is not available in the memory it returns NULL.

Syntax:

```
Char *str=(char *)malloc(40*sizeof(char));
```

•It creates 40 size block.

calloc()

•This function is used for allocates multiple block of memory.

•If the memory is not available in the memory it returns NULL.

•This is efficient than malloc for allocating space for multidimensional arrays.

•It initialize the blocks with zero(0).

Syntax:

Char *str=(char *)calloc(40,sizeof(char));

•It creates 40 individual blocks.

realloc()

•This function is used for reallocate a block of memory from a pointer we already have.

•It is used for shrink or expand the size of the memory.

Syntax:

```
Char *str=(char *)realloc(ptr,new_size);
```

```
main()
{
        char *names[6];
        char n[50];
        int len,I;
        char *p;
        for(i=0;i<=5;i++)
        {
                printf("\n enter any name:");
                scanf("%s",n);
                len=strlen(n);
                p=malloc((len)*sizeof(char));
                strcpy(p,n);
                names[i]=p;
        }
        for(i=0;i<6;i++)
        printf("\n %s",names[i]);
}
```

Dangling pointer:

If a pointer is pointing the memory address of an variable which is not alive that pointer called dangling pointer.

```
int *fun()
{
        int i=20;
        return (&i);
}
main()
{
        int *p;
        p=fun();
         fflush(stdin);
        printf("\n address is::%u\t value is::%d",&p,*p);
}
```

Output: **garbage value**

Memory leak:

```
#include<stdio.h>
#include<stdlib.h>
main()
{
        char *c=(char *)malloc(10*sizeof(char));
        char *c1=(char *)malloc(10*sizeof(char));
        gets(c);
        gets(c1);
        printf("\n %s",c);
        printf("\n %s",c1);
        c1=c;
        printf("\n %s",c);
        printf("\n %s",c1);
        free(c);
        free(c1);
}
```

DYNAMIC ARRAYS

Dynamic 1D array

```
#include<stdio.h>
#include<stdlib.h>
main()
{
        int *p,n,i;
        printf("\n how many elements you are going to enter::");
        scanf("%d",&n);
        p=(int *)malloc(n*sizeof(int));
        printf("\n enter elements::");
        for(i=0;i<n;i++)
        {
                scanf("%d“,(p+i));
        }
        printf("\n entered elements are follows::\n");
        for(i=0;i<n;i++)
        printf("%d\t",*(p+i));
        printf("\n");
}
```

Output:

```
 how many elements you are going to enter::3

 enter elements::1
2
3

 entered elements are follows::
1       2       3
```

Dynamic 2D array

```
main()
{
        int *p,m,n,val,i,j;
        printf("\n enter 2D array size::");
        scanf("%d%d",&m,&n);
        p=(int *)malloc(m*n*sizeof(int));
        printf("\n enter elements::");
        for(i=0;i<m;i++)
        for(j=0;j<n;j++)
        {
                scanf("%d“,(p+i*n+j));
        }
        printf("\n entered elements are follows::\n");
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                printf("%d\t%p\n",*(p+i*n+j),(p+i*n+j));
        printf("\n");
        }
        printf("\n");
}
```

Output:

```
enter 2D array size::2
2

 enter elements::1
2
3
4

 entered elements are follows::
1       151584776
2       151584780

3       151584784
4       151584788
```