DYNAMIC MEMORY ALLOCATION

- **Dynamic memory allocation in C:**
  - The process of allocation memory during program execution is called dynamic memory allocation.

- **Dynamic memory allocation functions in C:**
  - C language offers 4 dynamic memory allocation functions. They are,

  - malloc()
  - calloc()
  - realloc()
  - free()

Function	Syntax
malloc ()	malloc (number *sizeof(int));
calloc ()	calloc (number, sizeof(int));
realloc ()	realloc (pointer_name, number * sizeof(int));
free ()	free (pointer_name);

1. MALLOC() FUNCTION:

- **1. malloc() function:**
  - The name malloc stands for *memory allocation*.
  - malloc () function is used to allocate space in memory during the execution of the program.
  - malloc () does not initialize the memory allocated during execution. It carries garbage value.

[1] MALLOC()

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

void main()
{
        char *str;
        clrscr();
        str = (char *)malloc(15);
        strcpy(str,"computer");
        printf("String=%s , Address=%u",str,&str);
        free(str);
        getch();
}
```

2. CALLOC() FUNCTION

- 2. calloc() function:
  - The name calloc stands for <u>*contiguous allocation*</u>.
  - calloc () function is also like malloc () function. But calloc () initializes the allocated <u>memory to zero</u>. But, malloc() doesn’t.

[2] CALLOC ()

```
#include<stdio.h>  #include<conio.h> #include<alloc.h>
void main()
{
        int i,n;
        int *a;
        clrscr();
        printf("\n Number of elements to be entered:-");
        scanf("%d",&n);
        a = (int *)calloc(n,sizeof(int));
        printf("Enter %d number:\n",n);
        for(i=0;i<n;i++)
        {
                scanf("%d",&a[i]);
        }
        printf("The number entered are:\n");
        for(i=0;i<n;i++)
        {
                printf("\t%d",a[i]);
        }
        getch();
}
```

<u>3. realloc() function in C:</u>

- realloc () function modifies the allocated memory size by malloc () and calloc () functions to <u>new size</u>.

<u>4. free() function in C:</u>

- free () function frees the allocated memory by malloc (), calloc (), realloc () functions and <u>returns the memory to the system.</u>

[3] RELLOC ()

```
#include<stdio.h>   #include<conio.h>    #include<stdlib.h>
void main()
{
        char *str;
        clrscr();
        str = (char *)malloc(15);
        strcpy(str,"computer");
        printf("String=%s , Address=%u",str,&str);

        //Reallocating Memory
        str = (char *)realloc(str,25);
        printf("\n");
        strcat(str,".com");
        printf("String=%s , Address=%u",str,&str);
        free(str);
        getch();
}
```

[4] FREE()

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

void main()
{
        char *buffer;
        buffer = (char *)malloc(100);
        clrscr();
        strcpy(buffer,"Comp");
        printf("\n The buffer size is %d",*buffer);
        printf("\n Buffer contains %s",buffer);
        printf("\n Press any keys to free memory");
        free(buffer);
        getch();
}
```

Difference between static memory allocation and dynamic memory allocation:

Static memory allocation	Dynamic memory allocation
In static memory allocation, memory is allocated while writing the C program. Actually, user requested memory will be allocated at compile time.	In dynamic memory allocation, memory is allocated while executing the program. That means at run time.
Memory size can't be modified while execution. Example: array	Memory size can be modified while execution. Example: Linked list

DIFFERENCE BETWEEN MALLOC() AND CALLOC() FUNCTIONS :

malloc()	calloc()
It allocates only <u>single block of</u> requested memory	It allocates <u>multiple blocks</u> of requested memory
. Total = <u>80</u> bytes	Total = <u>1600</u> bytes
malloc () doesn't initializes the allocated memory. It contains garbage values	calloc () initializes the allocated memory to zero
type cast must be done since this function returns void pointer int *ptr; ptr = (int*)malloc(sizeof(int)*20);	Same as malloc () function int *ptr; ptr = (int*)calloc(20, 20 *