Dynamic Memory Management Functions :

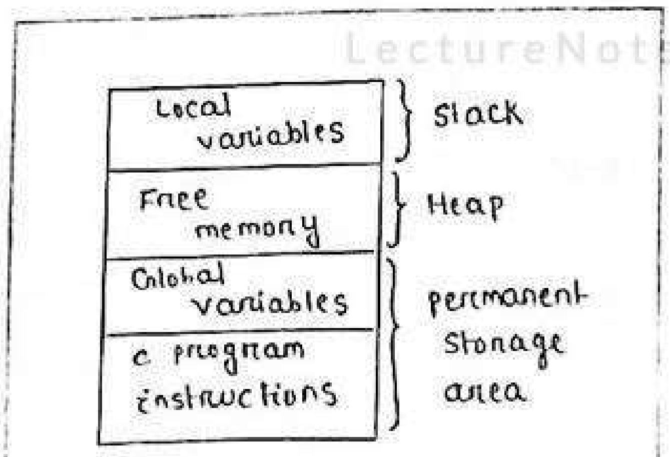
→ The process of allocating memory at run time is known as dynamic memory allocation.

→ In C, there are four library routines known as "memory management functions" that can be used for allocating and freeing memory during program execution. These are ;

- malloc : Allocates request size of bytes and returns a pointer to the first byte of the allocated space
- calloc : Allocates a space for an array of elements, initializes them to zero and then return a pointer to the memory.
- free : Frees previously allocated space
- realloc : Modifies the size of previously allocated space

→ These functions helps us to built complex application programs that use the available memory intelligently.

Memory allocation process :



(Storage of a c program)

→ The program instructions and global and static variables are stored in a region known as permanent storage area and the local variables are stored in another area called stack.

→ The memory space that is located between these two region is available for dynamic allocation during execution of the program. This free memory region is called heap.

→ The size of heap keeps changing when program is executed due to Creation and death of variables that are local to functions and blocks.

→ Therefore, it is possible to encounter memory "overflow" during dynamic allocation process. In such situation, the memory allocation functions (malloc, calloc, free, realloc) return a NULL pointer (when they fail to locate enough memory requested).

MALLOC : Allocating a block of memory

→ A block of memory may be allocated using the function malloc
→ The malloc function reserves a block of memory of specified size and returns a pointer of type void. That means, we can assign it to any type of pointer

→ Syntax :

```
ptr = (data-type *) malloc (n * sizeof(datatype));
 ↓                            ↓
is a pointer of             no. of element that is
type data-type              to be allocated
```

→ Example:

```
float *p;
p = (float *) malloc (10 * sizeof(float));
```

→ default - initial value : Garbage value

```
/* program to allocate a memory dynamically to store a matrix
   and print it */

# include <stdio.h>
# include <conio.h>
 void main()
 {
   int *p , i, j, r, c ;
   clrscr() ;
   printf (" Enter the order of matrix a:");
   scanf (" %d %d", &r , &c) ;

   p = (int *) malloc ( r*c * sizeof (int)) ;

 if ( p = = NULL)
        printf(" memory overflow") ;
 else
      {
          for (i=0 ; i<r ; i++)
             for( j=0 ; j<c ; j++)
               {
                 printf ("Enter a no") ;
                scanf (" %d", (p+ i*c+j)) ;
             }
    printf ( " The matrix is as follows:") ;

        for (i=0; i<r ; i++)
             {  for( j=0 ; j<c ; j++)
                  {
                   printf( " %d\t", *(p+i*c+j)) ;
                  }
               printf ("\n") ;
           }
     }
    getch() ;
  }
```

Q. Allocate memory dynamically for 'n' no of elements and print it.

```
#include <stdio.h>
# include <conio.h>
void main()
{
    float *p;
    int i, n;
    clrscr();
  printf(" Enter the no of elements :");
  scanf(" %d", &n);
    p = (float *) malloc (n * sizeof (float));
 if ( p = = NULL)
       printf(" Memory overflow or No availability of
                                        memory");
 else
     {
       for (i=0; i<n; i++)
          { printf(" Enter an eleoment :");
            scanf(" %d", (p+i));
          }
      printf(" Elements entered are :");
    for(i=0; i<n; i++)
       { printf(" %d", *(p+i));
       }
   getch();
}
```

CALLOC : Allocating multiple block of memory

→ Calloc is an another memory allocation function that is normally used for requesting memory space at run time for storing derived data types such as arrays and structures.

→ While malloc allocates a single block of storage space, calloc allocates a multiple block of storage, each of same size and then sets all bytes to zero.

→ The general form of calloc is :

```
Ptr = ( data-type * ) Calloc (n, sizeof (datatype));
 ↓                               ↓
is a pointer of              no. of element that is
 type data-type               to be allocated
```

→ Example : float *P ;
P = (float *) calloc (n, sizeof (float));

→ Default initial value : zero

/* program to allocate a memory dynamically to store a matrix and display it */

```
# include <stdio.h>
# include <conio.h>
void main()
 {
   int *p, i, j, r, c ;
   clrscr();
  printf(" Enter the order of matrix a:");
  scanf ("%d%d", &r ,&c) ;
  p = (int *) calloc ( r*c , sizeof (int));
```

→ It is not the pointer that is being released but rather what it points to.

→ To release an array of memory that was allocated by calloc we need only to release the pointer once. It is an error to attempt to release elements individually.

/* pgm to illustrate the free function */

```
# include <stdio.h>
# include <conio.h>
 void main()
 {
     int *p, i, j, r, c ;
      clrscr();
      printf(" Enter the order of matrix a:");
      scanf("%d %d", &r, &c);
   p = (int *) calloc (r*c, sizeof(int));

 if (p == NULL)
        printf(" Memory overflow");

 else
     {    for(i=0; i<r; i++)
              for (j=0; j<c ; j++)
                 {  printf(" Enter a no:");
                    scanf("%d",(p+i*c+j));
                 }
  printf(" The matrix is as follows:");

       for (i=0; i<r ;i++)
          {
            for (j=0; j<c; j++)
               {
                  printf("%d", *(p+i*c+j));
               }
```

```
    printf("\n");
  }
 }
  free(p);
 getch();
 }
```

REALLOC : Altering the size of a block

→ It is likely that, we discover later, the previously allocated memory is not sufficient and we need additional space for more elements.

→ It is also possible that the memory allocated is much larger than necessary and we want to reduce it

→ In both the cases, we can change the memory size already allocated with the help of the function realloc. This process is called the reallocation of memory.

→ Example : if the original allocation is done by the statement

ptr = malloc(size), then reallocation of space may be done by the statement;

ptr = realloc(ptr, newsize);

This function allocates a new memory space of size newsize to the pointer variable ptr and returns a pointer to the first byte of the new memory block. The new size may be larger or smaller than the size.

```
/* program to illustrate the realloc function */
# include <stdio.h>
# include <conio.h>
void main()
{ int *p, i, j, r, c, ~~ch~~, newsize;   char ch;
   clrscr();
 printf("Enter the order of matrix a:");
 scanf("%d%d", &r, &c);
p = (int *) Calloc (r*c, sizeof (int));
  printf(" would you want to change the size :");
   scanf("%c", &ch);
   fflush (stdin);
 if ( ch = = 'Y')
       {
        printf(" Enter newsize :");
         scanf("%d", &newsize);
 realloc ( p, newsize);
 }
if ( p = = NULL)
        printf(" Memory overflow");
 else {
          for(i=0; i<r; i++)
               for ( j=0; j<c; j++)
                  {
                    printf("Enter a no:");
                   scanf(" %d", (p+i*c+j));
                 }
printf(" The matrix is as follows : \n");
for(i=0; i<r ; i++)
   { for (j=0; j<c ; j++)
        { printf ("%d", *(p+i*c+j));
        }
     printf ("\n");
   }
}
  getch();
}
```