Functions for Dynamic Memory Allocation in C

- In Dynamic memory allocation, the memory is allocated at run time from the heap segment.
- We have four functions that help us achieve this task.
  - malloc()
  - calloc()
  - realloc()
  - free()

Malloc()

- malloc() stands for memory allocation.
- It reserves a block of memory with the given amount of bytes.
- The return value is a void pointer to the allocated space.
- therefore the void pointer needs to be casted to the appropriate type as per the requirements.
- If However, if the space is unsufficient, allocation of memory fails and it returns a NULL pointer.
- All the values at allocated memory are initialized to garbage values
- Syntax

```
ptr = (ptr-type *) malloc (size_in_bytes)
```

```
int *ptr;
ptr = (int *) malloc (3 * sizeof(int));
```

calloc() :-

- calloc() stands for contiguous allocation
- It reserves n blocks of memory with the given amount of bytes.
- the return value is a void point pointer to the allocated space.
- Therefore the void pointer needs to be casted to the appropriate type as per the requirements.
- However, if the space is unsufficient, allocation of memory fails and it returns a NULL pointer.
- All the values at allocated memory are initialized to 0.

Syntax

```
ptr = (ptr-type *) calloc (n, size-of-in-bytes);

int ptr * = (int *) calloc (3, sizeof(int));
```

Realloc () :-

- realloc() stands for reallocation.
- If the dynamically allocated memory is insufficient we can change the size of previously allocated memory using realloc() function.

Syntax

```
ptr = (ptr-type *) realloc (ptr, new-size-in-bytes);

ptr = (int *) realloc (ptr, 5 * sizeof(int))
```

free()

- free() is used to free the allocated memory
- If the dynamically allocated memory is not required anymore, we can free it using free function.
- This will free the memory using being used by the program in the heap.

Syntax

free (ptr)

Code

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

int main()
{
    // use of malloc
/*  int *ptr;
    int n;
    printf("Enter the size of the array you want to
            create \n");
    scanf("%d", &n);

    ptr = (int *) malloc (n * sizeof(int));
    for(int i=0 ; i<n; i++)
    {
        printf(" Enter the value of no %d of
                this array \n ", i)
        scanf( "%d", &ptr[i]);
    }
```

```
for (int i=0;i<n;i++)
{
    printf("The value of at %d of this array is
                                    %d \n", i, ptr[i]);
}  */

// use of calloc

int *ptr;
int n;
printf("Enter the size of the array you want to
                        create \n");
scanf("%d", &n);

ptr = (int *) calloc (&n, sizeof(int));

for (int i=0;i<n;i++)
{
    printf("Enter the value no %d of this array \n", i);
    scanf("%d", &ptr[i]);
}

for (int i=0;i<n;i++)
{
    printf("The value at %d of this array is %d \n",
                        i, ptr[i]);
}
```

```
// use of realloc

printf("Enter the size of the new array you want to
        create \n");
scanf("%d", &n);

ptr = (int *) realloc (ptr, n*sizeof(int));

for(int i=0 ; i<n; i++)
{
   printf("Enter the new value no %d of this array \n",i
   scanf(" %d ", &ptr[i]);
}

   for(int i=0;i<n;i++)
   {
      printf("The new value at %d of this array is
              %d \n", i, ptr[i]);
   }

   free(ptr);

   return 0;
}
```