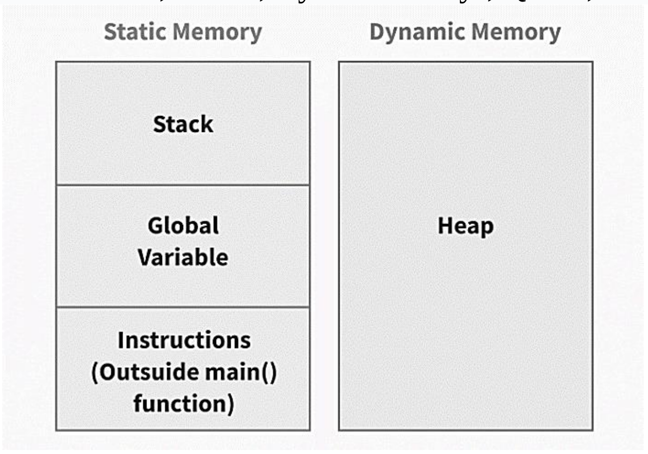
Practical-3
Dynamic Memory Allocation

The concept of **dynamic memory allocation in c language** *enables the C programmer to allocate memory at runtime*. Dynamic memory allocation in c language is possible by 4 functions of stdlib.h header file.

malloc()	allocates single block of requested memory.
calloc()	allocates multiple block of requested memory.
realloc()	reallocates the memory occupied by malloc() or calloc() functions.
free()	frees the dynamically allocated memory.

Dynamic Memory Allocation is a process in which we allocate or deallocate a block of memory during the run-time of a program. It can also be referred to as a procedure to use Heap Memory in which we can vary the size of a variable or Data Structure (such as an Array) during the lifetime of a program using the library functions. Dynamic Memory Allocation is considered as a very important concept in the field of Data Structures and is used in almost every Data Structures like Linked Lists, Stacks, Dynamic Arrays, Queue, etc.



malloc() function in C

- The malloc() function allocates single block of requested memory.
- It doesn't initialize memory at execution time, so it has garbage value initially.
- It returns NULL if memory is not sufficient.
- The syntax of malloc() function is given below:
  ptr=(cast-type*)malloc(byte-size)

calloc() function in C

- The calloc() function allocates multiple block of requested memory.
- It initially initialize all bytes to zero.
- It returns NULL if memory is not sufficient.
- The syntax of calloc() function is given below:
  ptr=(cast-type*)calloc(number, byte-size)

realloc() function in C

- If memory is not sufficient for malloc() or calloc(), you can reallocate the memory by realloc() function. In short, it changes the memory size.

- syntax of realloc() function.
  ptr=realloc(ptr, **new**-size)

free() function in C

- The memory occupied by malloc() or calloc() functions must be released by calling free() function. Otherwise, it will consume memory until program exit.
- syntax of free() function.
  free(ptr)

Example

```
// Program to calculate the sum of n numbers entered by the user
#include <stdio.h>
#include <stdlib.h>

int main() {
  int n, i, *ptr, sum = 0;

  printf("Enter number of elements: ");
  scanf("%d", &n);

  ptr = (int*) malloc(n * sizeof(int));

  // if memory cannot be allocated
  if(ptr == NULL) {
    printf("Error! memory not allocated.");
    exit(0);
  }

  printf("Enter elements: ");
  for(i = 0; i < n; ++i) {
    scanf("%d", ptr + i);
    sum += *(ptr + i);
  }

  printf("Sum = %d", sum);

  // deallocating the memory
  free(ptr);

  return 0;
}
```

Output:

```
Enter number of elements: 3
Enter elements: 100
20
36
Sum = 156
```

Exercise

1. Define a structure Book which has members that include book_name, author_name, price and pages. Create a structure pointer variable which takes book information from the user and print the book information for book_name that with ‘D’. Also create a function which display all book information using pointer.
2. Write a program in C to dynamically allocate memory using malloc function to store N integer numbers entered by the user and then print the sum of all elements. Also free memory at the end of program.
3. Write a program in C to reallocate previously allocated memory space. Print the address and value of original array and modified array [Take integer array]
4. Write a C program to accept a number n from the user and create an array of size n using dynamic memory allocation. Accept and store values in the array. Now take another number m from the user and revise the array size using dynamic memory allocation. Accept and store values in the revised array.
5. Write a program to store empID, name, age in a structure using DMA. The employee details should be stored for as many books as required. Display all the employee details.

```
//Use of calloc

#include <stdio.h>
#include <stdlib.h> //
int main()
{
    /*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;
}
```