PRACTICAL-2

Introduction to Dynamic Memory Allocation. DMA functions malloc(), calloc(), free() etc.

- **MALLOC ():**
- malloc () is used to allocate a fixed amount of memory during the execution of a program.
- malloc () allocates size in bytes of memory from heap, if the allocation succeeds, a pointer to the block of memory is returned else NULL is returned.
- Allocated memory space may not be contiguous.
- Each block contains a size, a pointer to the next block, and the space itself.
- The blocks are kept in ascending order of storage address, and the last block points to the first.
- The memory is not initialized. It initializes each block with default garbage value.

CODE:

```
#include<stdio.h>
#include<stdlib.h>
void main ()
{
int* ptr;
int n, i;
n = 5;
printf("\n ENTER NUMBER OF ELEMENTS:”);
scanf(“%d”, &n);
ptr = (int*) malloc (n * sizeof(int));
if (ptr == NULL)
    {
    printf("\n Memory not allocated ");
    }
else
   {
   printf("\n Memory Allocated Successfully");
       for (i = 0; i < n; i++)
         {
       ptr[i] = i + 1;
         }
      printf("\n THE ARRAY ELEMENTS ARE: ");
      for (i = 0; i < n; i++)
      {
      printf("%d, ", ptr[i]);
      }
  }
}
```

OUTPUT:

```
ENTER NUMBER OF ELEMENTS:5

Memory Allocated Successfully
THE ARRAY ELEMENTS ARE:12345
```

- **FREE ():**
- “free” method in C is used to dynamically de-allocate the memory.
- The memory allocated using functions malloc() and calloc() is not de-allocated on their own.
- Hence the free() method is used, whenever the dynamic memory allocation takes place.
- It helps to reduce wastage of memory by freeing it.

CODE:

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

void main ()
{
int* ptr;
int n, i;
n = 5;
printf("\n ENTER NUMBER OF ELEMENTS:”);
scanf(“%d”, &n);
ptr = (int*) malloc (n * sizeof(int));
if (ptr == NULL)
    {
    printf("\n Memory not allocated ");
    }
else
   {
   printf("\n Memory Allocated Successfully");
   free(ptr);
   printf(“Malloc memory deallocated successfully”);
    }
}
```

OUTPUT: