C Memory Management

C Memory Allocation and format is also

There are ways in which we can allocate memories dynamically in a heap. We use these four standard library functions often:

1. malloc():
malloc stands for memory allocation. This inbuilt function requests memory from the heap and returns a pointer to the memory. The pointer is of the void type and we can typecast it to any other data type of our choice. All the values at the allocation time are initialized to garbage values. The function expects the memory space along with the size we want in bytes at the time it is used.

Syntax:
ptr = (ptr - type *)malloc(size_in_bytes)

Example:
int *ptr;
ptr = (int *)malloc(5 * sizeof(int));

2. calloc():
calloc stands for contiguous memory allocation. Similar to malloc, this function also requests memory from the heap and returns a pointer to the memory. Differences lie in the way we have to call it. First, we have to send as parameters the number of blocks needed along with their size in bytes. Second, in calloc(), the values at the allocation time are initialized to 0 instead of garbage value unlike what happens in malloc().

Syntax:
ptr = (ptr - type *)calloc(n, size_in_bytes)

Example:
int *ptr;
ptr = (int *)calloc(5, sizeof(int));

3. realloc():
realloc stands for reallocation of memory. It is used in cases where the dynamic memory allocated previously is insufficient and there is a need of increasing the already allocated memory to store more data. We also pass the previously declared memory address, and the new size of the memory in bytes while calling the function.

Syntax:
ptr = (ptr - type *)realloc(ptr, new_size_in_bytes)

Example:
ptr = (int *)realloc(ptr, 10 * sizeof(int));

4. free():
While discussing the disadvantages of dynamic memory allocation, it was mentioned that there is no automatic deletion of dynamically allocated memory when the pointer gets overwritten. So, to manually do it, we use the free() function to free up the allocated memory space. Therefore, free() is used to free up the space occupied by the allocated memory. We just have to pass the pointer as a parameter inside the function and the address being pointed gets freed.

Syntax:

```
free(ptr);
```

C Memory Layout

What is Dynamic Memory?
Any allocation of memory space during the runtime of the program is called dynamic memory allocation. The concept of dynamic memory allocation is used to reduce the wastage of memory, and it is the optimal way of memory allocation.

Memory Allocation in C
Memory allocation in C can be divided into four segments.

1. Code:
Code composes of all the text segments of our program. Everything we do as a programmer to build a program falls into this category.

2. Variables:
Declarations of both global and static variables come into this segment. Global variables can be used anywhere in the program, while static has its limitations inside the function.

3. Stack:
A stack is a data structure. Initially, the stack looks like an empty bucket in which the last entry to be inserted will be the first one to get out. It is also known as a LIFO data structure, i.e., last in first out.

Suppose the program starts executing a function named A, then this function A gets pushed into the stack. Now, if function A calls another function B during its execution, then function B will also get pushed into the stack, and the program will start executing B. Now, if B calls another function C, then the program will push C into the stack and will start with its execution. After the program gets done with the execution of C, the program will pop C from the stack as it was the last one to get pushed and start executing B. When B gets executed completely, it will get popped and A will start executing further until the stack becomes empty.

4. Heap:
Heap is a tree-based data structure. It is used when we allocate memory dynamically. To use the heap data structure, we have to create a pointer in our main function that will point to some memory block in a heap. The disadvantage of using a heap is that the memory assigned to a pointer will not get freed automatically when the pointer gets overwritten.

Differences between static and dynamic memory

Static	Dynamic
Allocation of memory before execution	Allocation of memory at run time
Non-reusable memory	Reusable memory
Less optimal way	More optimal way

C Memory Allocation
There are ways in which we can allocate memories dynamically in a heap. We use these four standard library functions often:

1. malloc():
... (same explanation as above) ...
2. calloc():
... (same explanation as above) ...
3. realloc():
... (same explanation as above) ...
4. free():

... (same explanation as above) ...