Summary DSA

Sessions No 08(21-12-2022)

- **Stack Memory** :
  - ➔ Whenever a function is called, its variables get memory allocated on the stack. And whenever the function call is over, the memory for the variables is de-allocated.
  - ➔ It means allocation & de-allocation will happen automatically. We as a developer don't have control over that part.
  - ➔ As soon as the method finishes its execution all the data belonging to that method flushes out from the stack automatically.

- **Heap Memory** :
  - ➔ If we want to control then we have to used Heap memory.
  - ➔ It allows dynamic memory allocation, which is allocated at the time of execution of instructions.
  - ➔ Memory is allocated in the random order, and the pointers are used in order to access the data.
  - ➔ As a developer we have to manually deallocate the memory.

- If we want 4 bytes of space in memory then we have to use ***malloc()*** function in C.

  Eg. *int *p = malloc(4);*

  Where, p contains the address.

  And if we want to de-allocate then we have to use the ***free()*** function.

```c
#include <stdio.h>

// heap memory example
main() {
    int *p = malloc(4);  // allocation - 4 bytes
    *p = 50; // derefencing
    printf("%d", *p);
    free(p); // de-allocation
}
```

- Even though the function will stop , the heap memory data is still there if we don't deallocate memory using the free() function.

- **Memory Leakage** : When developer handles memory allocation and fails/forgot to free up the memory that is no longer required that time memory leakage will happen.

- If we want to allocate array in heap memory ie. dynamic array

```c
#include <stdio.h>

// heap memory example
main() {
    int *arr = (int *) malloc(5 * sizeof(int)); // allocation
    //here int * means we want to stored data int. By default it is int
    //5*sizeof(int)-means 5 data we want to stored and int size is 4 byte. Hence 5 * 4 = 20 byte we need
    *arr = 10; // stored 1st element
    *(arr + 1) = 20;  // stored second element
    printf("%d\n", *arr);
    printf("%d\n", *(arr+1));
    free(arr); // de-allocation
}
```

- In c++ dynamic memory allocation,

*int *p = new int;* // allocation

delete p; // de-allocation

If we want to create array size of 5 then,

*Int *arr = new int[5];*

- Mostly the new keyword means we want space from Heap memory.