NULL POINTER

A pointer that is assigned NULL is called a **null** pointer. It is used to initialize a pointer variable when that pointer variable isn't assigned any valid memory address yet.

If a pointer contains the null (zero) value, it is assumed to point to nothing

```
#include <iostream>

using namespace std;
int main () {
   int *ptr = NULL;
   cout << "The value of ptr is " << ptr;

   return 0;
}
```

When the above code is compiled and executed, it produces the following result −

```
The value of ptr is 0
```

Important Points

1. **NULL vs Uninitialized pointer –** An uninitialized pointer stores an undefined value. A null pointer stores a defined value, but one that is defined by the environment to not be a valid address for any member or object.
2. **NULL vs Void Pointer** – Null pointer is a value, while void pointer is a type

DANGLING POINTER

A pointer pointing to a memory location that has been deleted (or freed) is called dangling pointer.

De-allocation of memory

```
//Deallocating a memory pointed by ptr causes dangling pointer
#include <iostream>
int main()
{
    int *ptr = new int;
    // After below free call, ptr becomes a
    // dangling pointer
    delete ptr;
    // No more a dangling pointer
    ptr = NULL;
}
```

OUTPUT

```
No Output
```

WILD POINTER

A pointer which has not been initialized to anything (not even NULL) is known as wild pointer. The pointer may be initialized to a non-NULL garbage value that may not be a valid address.

```
int main()
{
    int *p; /* wild pointer */

    int x = 10;

    // p is not a wild pointer now
    p = &x;

    return 0;
}
```