**C Pointers**

**NULL Pointer**

You can give pointer a value equal to NULL (0) but you cannot use for read or write access

int \*p = NULL;

\*p = 8; // Write access not ok

printf(“%d”, \*p); // Read access, not ok

Compare with NULL

if (p == NULL) {}; // OK

Example - after calling malloc(), it is a best practice to check pointer returned for NULL

NULL pointer in DS

Link field to last node of a singly linked list

**Pointer to pointer**

char \*\*p; // **char \*** \*p

char \*p1;

p = &p1;

**Return pointer , pointer parameter (input, output)**

int \* func1(int \*, int, **int \***);

Returns a pointer to int

First parameter int \* can be for taking an input (Call by reference)

Second parameter int \* can be for returning second value (Call by reference)

Example – Array of student records

Return address of first record with score X **and** next lower score

**Subtract pointers** (Normally used when we have an array of …)

Two pointers must be pointing to members of same array

**pointer + 1** value is pointer value + sizeof (variable to which pointer points)

**1D Array and pointer**

Array name is a pointer with valid address – address of first element

int a[10];

a[2] = 5 ; or \*(a+2)=5;

**2D array and pointer**

int a[4][3];

a[1][2] same as \*(\*(a+1)+2)

**Array of pointers**

Good example

C command line argument argv

int main( int argc, char\*argv[])

{}

argv is an array of char pointers

**Dynamically allocate 2D array ( For 1D array we use calloc)**

2x3

int \*p[2];

use calloc (2 times) to allocate 1D array of 3 elements

Put pointer returned in p[0], p[1]

**Pointer to function**

See program attached