Assignment-2 (Pointers & Arrays): -

1. Create Pointers for various data types and test compatibility between them.
2. Usage of NULL pointer,try dereferencing NULL pointer.
3. Print equivalent bit pattern (in hexadecimal) for some float, double values.
4. Check the endianness (little or big) of your current system.
5. Conversion of short integer from little endian to big endian(network order) and vice versa.
6. Conversion of integer from little endian to big endian(network order) and vice versa.
7. Pointer airthmetic(try out for various data types) p=&x; p1=p+5; p2=p-5; p1++; p2--; p1-p2
8. Form equivalent expression for chain of pointers
   int x; int *p = &x; int **q = &p;
9. Given int a[5] = {10, 20, 30, 40, 50 }; int *p=a, *q=*(&a+1) - 1; evaluate following expressions
   *p++, *++p, (*p)++, ++(*p), ++*p, *(p++), *(++p)
   *q--, *--q, (*q)--, --(*q), --*q, *(q--), *(--q)
10. Convert from one type of pointer/address to other using void*
11. Test arithmetic operation on void pointers
12. Print all elements of a 1D array using a pointer, give equivalent expression for a[i] using pointers
13. Can we use a[i] or i[a] to access an element, test with some code
14. int arr[5]; int (*parr)[5]; parr=&arr; sizeof(parr), sizeof(*parr), sizeof(**parr) access array elements with suitable dereferencing of parr
15. Usage of assert macro before dereferencing any pointer.

16. Differentiate between the following declarations
    (a). #define PINT int*
    PINT p1, p2;
    (b). typedef int* pint
    pint p1, p2;
17. Differentiate betweenint *parr[5]; int (*parr) [5];
18. Differentiate between const int * p; int const * p int * const p = &x; const int * const p = &x;
    Try *p=20, p++, (*p)++, p=&y in each case

19. Test the following code const int x=10; int *p; p = &x; *p=20; printf("%d\n",x);
20. Access 2D array using pointers

    int arr[3][4]; int (*p)[4]; p=arr;

    sizeof(p), sizeof(*p), sizeof(**p) , values of p, p+1

    Check equivalence of arr[i][j], (*(p+i))[j], *(*(p+i)+j)
21. Store random numbers in an array and print them and perform liner search.
22. Give an expression to print last element of array irrespective of length using pointer notation.

    (You shouldn't consider length or size of array)
23. What is the significance of following pointer int (*q)[3][4];

    What are the sizes of q, *q, **q, ***q

    Write some code to test this with a 2D array.