Assignment-2 (Pointers): -

- Create Pointers for various data types and test compatibility between them.
- Usage of NULL pointer,try dereferencing NULL pointer.
- Print equivalent bit pattern (in hexadecimal) for some float, double values.
- Check the endianness (little or big) of your current system.
- Conversion of short integer from little endian to big endian(network order) and vice versa.
- Conversion of integer from little endian to big endian(network order) and vice versa.
- Pointer arithmetic (try out for various data types)

  p=&x; p1=p+5; p2=p-5;

  p1++; p2--; p1-p2
- Form equivalent expressions for chain of pointers

  int x; int *p = &x; int **q = &p;
- 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), --*p, *(q--), *(--p)
- Convert from one type of pointer/address to other using void*
- Test arithmetic operations on void pointers
- Print all elements of a 1D array using a pointer , give equivalent expression for a[i] using pointers
- Can we use a[i] or i[a] to access an element, test with some code
- int arr[5]; int (*parr)[5];

  parr=&arr;

  sizeof(parr), sizeof(*parr), sizeof(**parr)

  access array elements with suitable dereferencing of parr
- Usage of assert macro before dereferencing any pointer.

- Differentiate between the following declarations
  - #define PINT int*

    PINT p1,p2;
  - typedef int* pint

    pint p1,p2;
- Differentiate between
  - int *parr[5];
  - int (*parr)[5];
- 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
- Test the following code

  const int x=10; int *p;

  p = &x; *p=20; printf("%d\n",x);
- 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)
- Store random numbers in an array and print them and perform linear search.
- Give an expression to print last element of array irrespective of length using pointer notation. (You shouldn't consider length or size of array)
- 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