Lab **15**
Dynamic Memory Allocation (1D and 2D)

Dynamic memory allocation in C/C++ refers to performing memory allocation manually by programmer.

new operator:

The new operator denotes a request for memory allocation on the Free Store. If sufficient memory is available, new operator initializes the memory and returns the address of the newly allocated and initialized memory to the pointer variable.

Syntax to use new operator:

```
// Pointer initialized with NULL
// Then request memory for the variable
int *p = NULL;
p = new int;

            OR

// Combine declaration of pointer
// and their assignment
int *p = new int;
```

Initialize memory:

```
pointer-variable = new data-type(value);
Example:
int *p = new int(25);
float *q = new float(75.25);
```

Allocate block of memory:

```
Example:
        int *p = new int[10]
```

delete operator

Since it is programmer's responsibility to de-allocate dynamically allocated memory, programmers are provided delete operator by C++ language.

```
  delete p;
  delete q;
```

To free the dynamically allocated array pointed by pointer-variable, use following form of *delete*:

```
Example:
   // It will free the entire array
   // pointed by p.
   delete[] p;
```

If enough memory is not available in the heap to allocate, the new request indicates failure by throwing an exception of type std::bad_alloc, unless “nothrow” is used with the new operator, in which case it returns a NULL pointer. Therefore, it may be good idea to check for the pointer variable produced by new before using it program.

```
int *p = new(nothrow) int;
if (!p)
{
   cout << "Memory allocation failed\n";
}
```

Following is a simple example demonstrating DMA in single dimensional array.

```
#include <iostream>

#define N 10

// Dynamically Allocate Memory for 1D Array in C++
int main()
{
    // dynamically allocate memory of size N
    int* A = new int[N];

    // assign values to allocated memory
    for (int i = 0; i < N; i++)
        A[i] = i + 1;

    // print the 1D array
    for (int i = 0; i < N; i++)
        std::cout << A[i] << " ";    // or *(A + i)

    // deallocate memory
    delete[] A;

    return 0;
}
```

Following is a simple example demonstrating DMA in 2 dimensional array.

```cpp
#include <iostream>

// M x N matrix
#define M 4
#define N 5

// Dynamic Memory Allocation in C++ for 2D Array
int main()
{
    // dynamically create array of pointers of size M
    int** A = new int*[M];

    // dynamically allocate memory of size N for each row
    for (int i = 0; i < M; i++)
        A[i] = new int[N];

    // assign values to allocated memory
    for (int i = 0; i < M; i++)
        for (int j = 0; j < N; j++)
            A[i][j] = rand() % 100;

    // print the 2D array
    for (int i = 0; i < M; i++)
    {
        for (int j = 0; j < N; j++)
            std::cout << A[i][j] << " ";

        std::cout << std::endl;
    }

    // deallocate memory using delete[] operator
    for (int i = 0; i < M; i++)
        delete[] A[i];

    delete[] A;

    return 0;
}
```

Tasks

Problem 1:
Declare an array of 10 elements dynamically and initialize each element (from user). Make another variable dynamically and take input from the user . Now, tell user whether that number is present in array or not.

Problem 2:
Take 20 integer inputs from user (all dynamically) and print the following:
number of positive numbers
number of negative numbers
number of odd numbers
number of even numbers

Problem 3:
Write a C++ program to find the largest element of a given array of integers (all dynamically), you can either ask user to enter the values or set the values yourself.

Problem 4:
Write a program that takes rows and cols from user an dynamically declare a 2D array. Initialize the array and then find the euclidean norm of the matrix.

Problem 5:
Modify problem 4 in such a way that you have to make a function which takes the 2D array as an argument, calculates the norm and return the result to main as output.