# CL-217
# Object Oriented Programming

**Objectives:**

- Searching array Methods
- Sorting array Methods
- Pointers
- Pointer Variable Declarations and Initialization
- Referencing/Dereferencing, Pointer Arithmetic Pointers & Functions

**Note: Carefully read the following instructions (*Each instruction contains a weightage*)**

1. There must be a block of comments at start of every question's code by students; the block should contain brief description about functionality of code.
2. Comment on every function about its functionality.
3. Use understandable name of variables.
4. Proper indentation of code is essential
5. Write a C++ statement(s) for each of the following task one after the other, in the same order.
6. Make a Microsoft Word file and paste all of your C++ code with all possible screenshots of every **task outputs in MS word and do not submit .cpp file with word file**.
7. First think about statement problems and then write/draw your logic on copy.
8. After copy pencil work, code the problem statement on MS Studio C++ compiler.
9. At the end when you done your tasks, attached C++ created files in MS word file and make your submission on Microsoft teams. (Make sure your submission is completed).
10. Please submit your file in this format 19F1234_L3.
11. Do not submit your assignment after deadline.
12. Do not copy code from any source otherwise you will be penalized with negative marks.

**Problem 1: | Pointers and file handling**

Declare a character array cArray of length 10.

1. Input the values from user.
2. Pass that array to a function copy using a pointer.
3. Copy the values of cArray in c1Array using dereferencing.
4. Note that c1Array is declared in function copy. Stored the c1Array in text file name Copy.txt using pointer variable.

**Problem 2: | Dynamic Memory Allocation**

Define an **`int*`** pointer variable **`a`**. Then:

1. Use **new** to make **a** point to a dynamic array of **5** cells of type **int**.
2. Write a loop to fill **a** with values 3, 7, 11, 15, 19.
3. Using **Print function** to print values stored in **a**.

**Problem 3: | Dynamic Memory Allocation**

Suppose you are developing a program that works with arrays of integers, and you find that you frequently need to duplicate the arrays. Rather than rewriting the array-duplicating code each time you need it, you decide to write a function that accepts an array and its size as arguments, creates a new array that is a copy of the argument array, and returns a pointer to the new array. The function will work as follows:

1. Accept an array and its size as arguments.
2. Dynamically allocate a new array that is the same size as the argument array.
3. Copy the elements of the argument array to the new array.
4. Return a pointer to the new array.
5. Release memory to memory heap
6. Int * copyFun(int arr[], int SIZE)

**Program Output Here are the**
Original array contents:
100 200 300 400 500 10 20 30 40 50 60
Here are the duplicate arrays:
100 200 300 400 500 10 20 30 40 50 60

**Problem 4: | Dynamic Memory Allocation and file handling**

A file contains the names of 'N' number of students. You have to perform the following tasks for the file.

1. Read the name of each student from the file inputFile.txt and store it in a dynamic array. Separate each name using "," comma character.
2. Print all the names in an output file named toOutputA.txt. Only one name store in single line.
3. Now print the names in reverse order in an output file named toOutputB.txt.
4. Release the allocated memory to memory heap. Memory leaks must be avoided

**Problem 5: | Dynamic Memory Allocation**

Write a program that dynamically allocates an array large enough to hold a user defined number of test scores. Once all the scores are entered, the array should be passed to a function that sorts them in ascending order. Another function should be called that calculates the average score. The program should display the sorted list of scores and averages with appropriate headings. Use pointer notation rather than array notation whenever possible.

**Note** Do not accept negative numbers for test scores.

**Problem 6: | Dynamic Memory Allocation**

Write a function named getString that has a local 2D char array of 100 elements. The function should ask the user to enter a sentence, and store the sentence in the array. Then the function should dynamically allocate a char array just large enough to hold the sentence, plus the null terminator. It should copy the sentence to the dynamically allocated array, and then return a pointer to the array. Demonstrate the function in a complete program.

char *copySen(char *ptr,int size);

**Problem 7: | (Double pointers, DMA, Passing pointers as parameter, Returning pointers from function)**

Write a C++ program to add 2 matrices A and B. Matrices should have user defined size. Use double pointers and dynamic memory allocation to create matrices. If both have equal size, then result of A+B should be saved in C matrix otherwise terminate the program. Write the following functions

- Void Input(int** p, int row, int col) // this function will input the p matrix
- Void Display(int** p, int row, int col) // this function will output the p matrix to console
- Int** Sum(int** p, int row, int col, int ** q , int row2, int col2) // this function will take sum of two matrices and return the resultant matrix.
- Print the resultant matrix in main using Display() function.
- Display A , B and C in matrix on console.

**Example**

A = 1 2

3 4

B = 6 5

7 1

C = 7 6

10 5

**Problem 8: | : (Double pointers, Dynamic Memory Allocation)**

Write a C++ program that will take input of 1D dynamic array named as Matrix. Now you will treat this array as 2D Matrix. User will input the size of array. Make sure it should be a square matrix otherwise take the input again.

Let's Say we have the following array of size=16 with row_size=column_size=4.

Matrix: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Its matrix representation will be

Matrix:

1 2 3 4

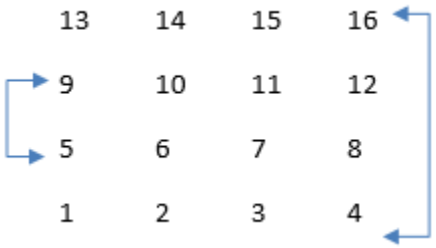
5 6 7 8

9 10 11 12

13 14 15 16

Now your task is to call a function SwapIt(Matrix) to swap its first row with last row, 2nd row with 2nd last row (and so on for matrix with greater size). After this function call DisplayMatrix().

Now your task is to call a function SwapIt(Matrix) to swap its first row with last row, 2nd row with 2nd last row (and so on for matrix with greater size). After this function call DisplayMatrix().




Write your program for integer array named Matrix of size=16 with row_size=4 but your program should be generic enough that if user enter the size=25 with row_size=5 or any other size it should work fine. No other array should be used for this task. Your main should be in following flow.

Int* Matrix;

Input(int* Matrix, int size);

MatrixDisplay(int* Matrix, int size); // this function will display the Matrix array in matrix form

SwapIt(int* Matrix, int size); // this function will swap the rows

MatrixDisplay(int* Matrix, int size); // Now this function will display the Matrix array in matrix form with swapped row

**Problem 4: (Double pointers, Dynamic Memory Allocation)**

Write a C++ program to build a matrix that have different number of elements in each row (different number of column in each row) using two-dimensional dynamic array. For Example

```
Enter the number of rows=3
Enter the number of col in row 1 =3
Enter 3 elements in row 1 =1 2 3
Enter the number of col in row 2 =5
Enter 5 elements in row 2 =4 5 6 7 8
Enter the number of col in row 3 =2
Enter 2 elements in row 3 =1 2

Matrix is
1  2  3
4  5  6  7  8
1  2
```

Your program must contain two functions. One for filling the elements into your two dimensional array and other for printing that array or matrix.

Proper code indentation will hold extra marks !

Best of luck ☺

**You are done with your exercise, submit on Teams at given time.**