Data Structures & Algorithm
LAB#01

Activity Outcomes:

This lab will give you the revision of **Programming Fundamentals** in C++ language.

- Revision to **pointers.**
- Revision to **character Arrays.**

Task 1: Memory Representation

Make a picture of memory of this program and find output of program.

```
#include <iostream>
using namespace std;

int main ()
{
  int firstValue = 5, secondValue = 15;
  int * p1, * p2;

p1 = &firstValue;
p2 = &secondValue;
*p1 = 10;
*p2 = *p1;
p1 = p2;
*p1 = 20;

  cout << "firstvalue is " << firstValue << '\n';
  cout << "secondvalue is " << secondValue << '\n';
  return 0;
}
```

Task 2: Pointer value vs Data value

Introduce **int** variables **x** and **y**, initialize them with 5 and 10 respectively. Then create **int*** pointer variables **p** and **q**. Point **p** to the address of **x**, and **q** to the address of **y**, then swap the values of **x** and **y** using pointers (**p** and **q**).

Then print the following information:

(1) The address of x and the value of x.
(2) The value of p and the value of *p.
(3) The address of y and the value of y.
(4) The value of q and the value of *q.
(5) The address of p (not its contents!).
(6) The address of q (not its contents!)

Task 3: Pointer Arithmetic

1. Introduce 2 variables i (int), j (float) and initialize them with 5 and 1.5 respectively.
2. Introduce 2 variables xPtr (int*), yPtr (float*) and assign address of i, j respectively.
3. Print values of i, j and also print addresses of xPtr, yPtr.
4. xPtr ++; yPtr ++;
5. Print addresses of xPtr, yPtr.
6. xPtr --; yPtr --;
7. Print addresses of xPtr, yPtr.
8. xPtr = xPtr + 3; yPtr = yPtr + 4;
9. Print addresses of xPtr, yPtr.

Task 4: Dynamic Memory

Define pointer variables of type (int*, char*, float*) and point to dynamic memory to store integer, character and floating point number. Input values for each type by user and show entered values.

Note: *Also check whether memory is assigned successfully or not.*

```
If ( !(iptr == new int ))
 {
        cout << "Error: out of memory.";
 }
```

Delete allocated dynamic memory at end. After “Delete”, print pointer values and values to which they are pointing.

Task 5: Memory Leak and Dangling Pointers

Run this code and check if there is any issue, also detect dangling pointer, memory leak and illegal memory access.

```
int*ptr = new int;
cout << "Enter Int Value: ";
cin >> *ptr;
cout <<"Value is: " <<*ptr << endl;
cout << "Pointer Value is: " << ptr << endl;
delete ptr;
cout << "After Del, Value is: " << *ptr << endl;
cout << "After Del, Pointer Value is: " << ptr << endl;
cout << "Dangling Pointer? If Yes, then Resolve issue" << endl;
cout << "Dynamically occupied Float Variable: "<<new float << endl;

int*ptr1 = new int;
*ptr1=9;
cout << *ptr1 << endl;
ptr1 = new int;
cout << *ptr1 << endl;
*ptr1=9;
cout << "Is there any Memory Leak" << endl;

int a;
int *ptr2;
*ptr2 = 10;
// if any illegal access, resolve it.
```

Task 6: Pointer passing in a function

Write two functions as shown below. These function will receive two variables as arguments and return nothing.

1. Swap => receive 2 integers, swap them, but this swap will not be visible in main
2. SwapByPointer => receive 2 integer pointers, swap their values, but this swap will not visible in main.

Note: In “main” call all two functions one by one, by passing 2 variables.

Task 7: Most Frequent Character

Write a C++ Program that takes a string as input and finds the most frequent character in the given string.

Example:

Input: str = "lalalalalalallalal";

Output: Most frequent character is 'l'

Task 8: String Comparison

Write a C++ program that takes two strings as input and checks if the two strings are equal or not.

Examples:

Input: ABCD, XYZ

Output: ABCD is not equal to XYZ

Input: DATA, data

Output: DATA is not equal to data

Input: DATA, DATA

Output: DATA is equal to data