C-DAC Mumbai
Assignment.1)

1. Write a program to declare a pointer, initialise it with the address of a variable, and
print the value using both the pointer and the variable. Demonstrate pointer assignment using two integer variables.
->

```
#include <iostream>
using namespace std;
int main()
{
    int a=10, b=20;
    int *ptr = &a;

    cout << "Value of a: "<<a<<endl;
    cout << "Value using pointer: "<< *ptr <<endl;

    ptr = &b;

    cout << "Value of b: "<<b<<endl;
    cout << "Value using pointer: "<< *ptr <<endl;

    return 0;
}
```

2. Write a program that explains the concept of a wild pointer and how it can lead to
undefined behaviour. Show how initialising a pointer can resolve this issue.
->

```
#include <iostream>
using namespace std;
int main()
{
  int *wildPtr;
  int a=30;
  wildPtr = &a;
  cout << "Value of a: "<< *wildPtr<< endl;
  return 0;
}
```

3. Create a program to demonstrate the use of NULL and its importance in pointer initialisation. Write code to check for NULL before dereferencing a pointer.
->

```
#include <iostream>
```

```
using namespace std;
int main()
{
    int *ptr = NULL;

    if (ptr)
    {
        cout << "Pointer value: " << *ptr << endl;
    }
    else
    {
        cout << "Pointer has value NULL" << endl;
    }
    return 0;
}
```

4. Write code to show the behaviour of pointers with const qualifier in various scenarios:
i. Pointer to a const value.
ii. const pointer to a value.
iii. const pointer to a const value.
->

```
#include <iostream>
using namespace std;
int main()
{
 int x=10, y=20;
 //first
    const int *ptr1 = &x;
    ptr1 = &y;

//second
    int *const ptr2 = &x;
    *ptr2 = 15;

//third
    const int *const ptr3 = &x;

    return 0;
}
```

5. Write a program demonstrating the difference between const int *ptr, int *const ptr, and
const int *const ptr.
->

```
#include <iostream>
using namespace std;
int main()
```

```
{
    int x = 30, y = 40;

    const int *ptr1 = &x;
    int *const ptr2 = &x;
    const int *const ptr3 = &x;

     ptr1 = &y;
    *ptr2 = 30;

    return 0;

}
```

6. Create a program that demonstrates how type-casting a const pointer can lead to unexpected behaviour.
->

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

int main() {
    const int a = 10;
    const int *ptr = &a;

    int *newPtr = const_cast <int *> (ptr);
    *newPtr = 20; // Modifying const value (undefined behavior)

    cout << "Value of a: " << a << endl;
    cout << "Modified value: " << *newPtr << endl;

    return 0;
}
```

7. Write a short program in both C and C++ that declares a structure, initializes it, and
prints its members.
->

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

    struct point {
        int x,y;
    };

    int main() {
        point p = {50,100};
        cout << "Point: (" << p.x << ", " << p.y << ")" << endl;
        return 0;
```

```
}

//#include <stdio.h>
//struct Point {
//    int x, y;
//};
//
//int main() {
//    struct Point p = {10, 20};
//    printf("Point: (%d, %d)\n", p.x, p.y);
//    return 0;
//}
```

8. Create a struct in C++ and add member functions to initialize data members and display their values.

->

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

struct Student {
    string name;
    int age;

    void initialize(string n, int a) {
        name = n;
        age = a;
    }

    void display() {
        cout << "Name: " << name << ", Age: " << age << endl;
    }
};

int main() {
    Student s;
    s.initialize("Yugandhar", 23);
    s.display();
    return 0;
}
```

9. Write a program to declare an array of structures to store information about 5 students
(e.g., Name, Age, Marks). Allow the user to input details and print the list.

->

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

struct Student {
```

```
    string name;
    int age;
    float marks;
};

int main() {
    Student students[5];

    for (int i = 0; i < 5; i++) {
        cout << "Enter name, age, marks for student " << i + 1 << ": ";
        cin >> students[i].name >> students[i].age >> students[i].marks;
    }

    for (int i = 0; i < 5; i++) {
        cout << "Student " << i + 1 << ": " << students[i].name << ", " <<
students[i].age << ", " << students[i].marks << endl;
    }

    return 0;
}
```

10.Write a C program that uses typedef to define a struct for a 2D point (x, y) and performs operations like distance calculation between two points.

->

```
#include <stdio.h>
#include <math.h>
typedef struct Point {
    int x, y;
} Point;

float distance(Point p1, Point p2) {
    return sqrt(pow(p2.x - p1.x, 2) + pow(p2.y - p1.y, 2));
}

int main() {
    Point p1 = {0, 0}, p2 = {8, 15};
    printf("Distance: %.2f\n", distance(p1, p2));
    return 0;
}
```

11. Create a C++ program that declares a class with public, private, and protected access specifiers. Demonstrate how access specifiers control access to members.

->

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

class Example {
```

```
public:
    int publicVar;

private:
    int privateVar;

protected:
    int protectedVar;

public:
    Example() {
        publicVar = 10;
        privateVar = 20;
        protectedVar = 30;
    }

    void display() {
        cout << "Public: " << publicVar << ", Private: " << privateVar << ",
Protected: " << protectedVar << endl;
    }
};

int main() {
    Example ex;
    ex.display();
    ex.publicVar = 15;
    // ex.privateVar = 25;
    // ex.protectedVar = 35;
    return 0;
}
```

12.Write a program to create a class called Employee with the data members name, id, and salary. Implement member functions to initialize and display data. Create multiple
objects to show how the class works.
->

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

class Employee {
    string name;
    int id;
    float salary;

public:
    void initialize(string n, int i, float s) {
        name = n;
        id = i;
        salary = s;
```

```
    }

    void display() {
        cout << "Name: " << name << ", ID: " << id << ", Salary: " << salary <<
endl;
    }
};

int main() {
    Employee e1, e2;
    e1.initialize("Yugandhar", 101, 50000);
    e2.initialize("Gaurav", 102, 60000);

    e1.display();
    e2.display();
    return 0;
}
```