1 Write a program to show the base of a numeric value of a variable using Hex, Oct and Dec manipulator functions.

```
#include <iomanip>
 using namespace std;
 int main ()
{
 int value;
 cout << "Enter number" << endl;
 cin >> value;
 cout << "Decimal base = " << dec << value << endl;
 cout << "Hexadecimal base = " << hex << value << endl;
 cout << "Octal base = " << oct << value << endl;
 return 0;
 }
```

2 Write a program to Find least common multiplier (LCM).

```
#include <iostream>
using namespace std;
int main() {
 int a=42, b=63, lcm;
 if(a>b)
 lcm = a;
 else
 lcm = b;
 while(1) {
if( lcm%a==0 && lcm%b==0 ) {
 cout<<"The LCM of "<<a<<" and "<<b<<" is "<<lcm;
 break;
 }
 lcm++;
 }
 return 0;
}
```

3 Write a CPP Program to find the area of a rectangle, a triangle and surface area of a sphere using function overloading.

```
#include <iostream>
#include <cmath>
using namespace std;
double area(double length, double width) {
    return length * width;
}
double area(double base, double height, int) {
    return 0.5 * base * height;
}
double area(double radius) {
    return 4 * M_PI * radius * radius;
}
int main() {
    double length, width, base, height, radius;
    cout << "Enter length and width of the rectangle: ";
    cin >> length >> width;
```

```
    cout << "Area of the rectangle: " << area(length, width) << endl;
    cout << "Enter base and height of the triangle: ";
    cin >> base >> height;
    cout << "Area of the triangle: " << area(base, height, 0) << endl;
    cout << "Enter radius of the sphere: ";
    cin >> radius;
    cout << "Surface area of the sphere: " << area(radius) << endl;
    return 0;
}
```

4 Write a CPP Program to create constructor with arguments and pass the arguments to Constructor.

```
#include <iostream>
using namespace std;
class Rectangle {
    double length, width;
public:
    Rectangle(double l, double w) : length(l), width(w) {}
    double area() { return length * width; }
};
int main() {
    double l, w;
    cout << "Enter the length and width of the rectangle: ";
    cin >> l >> w;
    Rectangle rect(l, w);
    cout << "The area of the rectangle is: " << rect.area() << endl;
    return 0;
}
```

5 Write a program by applying multiple and multilevel inheritance concepts.

```
#include <iostream>
using namespace std;
class Animal {
public:
    void eat() { cout << "Eating..." << endl; }
};
class Mammal {
public:
    void walk() { cout << "Walking..." << endl; }
};
class Dog : public Animal, public Mammal {
public:
    void bark() { cout << "Barking..." << endl; }
};
class Puppy : public Dog {
public:
    void weep() { cout << "Weeping..." << endl; }
};
int main() {
    Puppy myPuppy;
    myPuppy.eat();
    myPuppy.walk();
    myPuppy.bark();
```

```
    myPuppy.weep();
    return 0;
}
```

6 Write a program using polymorphism and virtual function.

```
#include <iostream>
using namespace std;
class Animal {
public:
    virtual void sound() { cout << "Some animal sound" << endl; }
};
class Dog : public Animal {
public:
    void sound() override { cout << "Bark" << endl; }
};
class Cat : public Animal {
public:
    void sound() override { cout << "Meow" << endl; }
};
void makeSound(Animal* animal) { animal->sound(); }
int main() {
    makeSound(new Dog());
    makeSound(new Cat());
    return 0;
}
```

7 Write a program to illustrate dynamic allocation and deallocation of memory using the new and delete operator.

```
#include <iostream>
using namespace std;
int main() {
    int* num = new int(42);
    cout << "Dynamically allocated integer: " << *num << endl;
    int size = 5;
    int* arr = new int[size];
    for (int i = 0; i < size; ++i) arr[i] = i + 1;
    cout << "Dynamically allocated array: ";
    for (int i = 0; i < size; ++i) cout << arr[i] << " ";
    cout << endl;
    delete num;
    delete[] arr;
    return 0;
}
```

8 Write a program for file Handling using ifstream & ofstream classes.

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
    ofstream outFile("example.txt");
    if (!outFile) return 1;
    outFile << "Hello, World!" << endl
```

```
        << "This is a sample file." << endl
        << "File handling in C++ is easy." << endl;
    outFile.close();
    ifstream inFile("example.txt");
    if (!inFile) return 1;
    string line;
    while (getline(inFile, line)) cout << line << endl;
    inFile.close();
    return 0;
}
```

9 Write a Program to implement the Bubble sort using template function.

```
#include <iostream>
using namespace std;
template <class T> void bubbleSort(T a[], int n)
{
 for (int i = 0; i < n - 1; i++)
 for (int j = n - 1; i < j; j--)
 if (a[j] < a[j - 1])
 swap(a[j], a[j - 1]);
}
int main()
{
 int a[5] = { 10, 50, 30, 40, 20 };
 int n = sizeof(a) / sizeof(a[0]);
 bubbleSort<int>(a, n);
 cout << " Sorted array : ";
 for (int i = 0; i < n; i++)
 cout << a[i] << " ";
 cout << endl;
 return 0;
}
```

10 Write a program containing a possible exception. Use a try block to throw it and catch block to handle it properly

```
#include <iostream>
using namespace std;
int main() {
    int a = 10, b = 0;
    try {
        if (b == 0) throw "Division by zero error!";
        cout << "Result: " << a / b << endl;
    }
    catch (const char* e) {
        cout << "Exception caught: " << e << endl;
    }
    cout << "Program continues after the exception handling." << endl;
    return 0;
}
```