# Task 4: Virtual Functions Usage (Polymorphism in C++)

Objective:
The objective of this task is to demonstrate runtime polymorphism in C++ using virtual functions. By using a base class pointer to call overridden methods of derived classes, the concept of dynamic binding is clearly illustrated.

## Problem Statement

Implement polymorphism using virtual functions by performing the following steps:
- Create a base class Shape with a virtual function draw()
- Derive Circle and Square classes from Shape
- Override the draw() function in derived classes

## Concepts Used

- Virtual Functions
- Runtime Polymorphism
- Inheritance
- Function Overriding
- Dynamic Binding

## Source Code (C++)

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

// Base class
class Shape {
public:
        virtual void draw() {
        cout << "Drawing a generic shape." << endl;
        }
};

// Derived class Circle
```

```
class Circle : public Shape {
public:
        void draw() override {
        cout << "Drawing a Circle." << endl;
        }
};

// Derived class Square
class Square : public Shape {
public:
        void draw() override {
        cout << "Drawing a Square." << endl;
        }
};

int main() {
        Shape* shape;

        Circle c;
        Square s;

        shape = &c;
        shape->draw();

        shape = &s;
        shape->draw();

        return 0;
}
```

## Explanation of the Code

1. A base class Shape is created with a virtual function draw().
2. The draw() function is marked virtual to enable runtime polymorphism.
3. Circle and Square classes inherit from Shape.
4. Both derived classes override the draw() function with their own implementations.
5. A base class pointer (Shape*) is used in main().
6. The base class pointer points to derived class objects.
7. The correct draw() function is called at runtime based on the object type.

## Program Output

```
student@student-virtual-machine:~/25SUB4508_LSP/25SUB4508_56133/ClassWork/day18$ g++ virtual_functions_polymorphism.cpp
student@student-virtual-machine:~/25SUB4508_LSP/25SUB4508_56133/ClassWork/day18$ ./a.out
Drawing a Circle.
Drawing a Square.
student@student-virtual-machine:~/25SUB4508_LSP/25SUB4508_56133/ClassWork/day18$
```

## Conclusion

This task demonstrates runtime polymorphism in C++ using virtual functions. By calling overridden functions through a base class pointer, the program ensures correct method execution at runtime. Virtual functions are essential for achieving dynamic behavior in object-oriented design.