FACULTY OF COMPUTER SCIENCE AND ENGINEERING
Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Topi

Lab Duration: 3 hrs. **CS112L Object Oriented Programming** **Marks: 10**

Lab No: 10 **Instructor: Engr. Hifza Umer** **Dated: 25/04/2023**

same funct behaving differently in diff dass.

Task 1.

Write a C++ program that demonstrates the usage of the typeid operator with polymorphic classes. Create two classes: Shape and Rectangle, where Rectangle is derived from Shape. Each class should have a virtual function printType() that prints the type of the object using typeid.

provides a program with the ability to retrive the actual derived type of obj

Instructions:

- Define a base class Shape with a virtual function printType().
- Define a derived class Rectangle that inherits from Shape and overrides the printType() function.
- In the main() function, create a pointer to Shape and a pointer to Rectangle, and instantiate objects of each class.
- Use typeid to print the type of each pointer and each object.

referred to by a pointer or a reference.

Expected Output:

```
Type of shapePtr: P5Shape
Type of rectanglePtr: P9Rectangle
Type of shapeObj: 5Shape
Type of rectangleObj: 9Rectangle
```

OR

It gives info of the obj.

class that cannot be used to make objects

Task 2.

Write a C++ program to simulate a banking system. Implement three classes: Account, SavingsAccount, and CheckingAccount. The Account class should be an abstract base class with a virtual function displayBalance() to display the balance. The SavingsAccount class should inherit from Account and implement the displayBalance() function to display the balance along with the interest earned. The CheckingAccount class should also inherit from Account and implement the displayBalance() function to display the balance along with the number of checks written.

In the main() function, create objects of type SavingsAccount and CheckingAccount. Simulate transactions such as deposits, withdrawals, writing checks, and earning interest. Use dynamic casting to ensure that the appropriate version of the displayBalance() function is called based on the type of account.

Instructions:

- Define a base class Account with a pure virtual function displayBalance().
- Define a derived class SavingsAccount that inherits from Account and implements the displayBalance() function to display the balance along with the interest earned.
- Define a derived class CheckingAccount that inherits from Account and implements the displayBalance() function to display the balance along with the number of checks written.
- In the main() function, create objects of type SavingsAccount and CheckingAccount, and perform transactions such as deposits, withdrawals, and check writing.
- Use dynamic casting to ensure that the appropriate version of the displayBalance() function is called based on the type of account.

Expected Output:

```
Savings Account Balance: $1500.00
Interest Earned: $15.00
Checking Account Balance: $2000.00
Checks Written: 0
Checking Account Balance: $1900.00
Checks Written: 1
```