# Assignment 1

Take a look at this example code:

#include <stdio.h>

class Shape {

public:

virtual ~Shape();

virtual void draw() = 0;

};

class Circle : public Shape {

public:

virtual ~Circle();

virtual void draw();

};

Shape::~Shape() {

printf("shape destructor\n");

}

// void Shape::draw() {

// printf("Shape::draw\n");

// }

Circle::~Circle() {

printf("circle destructor\n");

}

void Circle::draw() {

printf("Circle::draw\n");

}

int main() {

Shape \*shape = new Circle;

shape->draw();

delete shape;

return 0;

}

Verify your understanding of how the virtual keyword and method overriding work by performing a few experiments:

1. Remove the virtual keyword from each location individually, recompiling and running each time to see how the output changes. Can you predict what will and will not work?
2. Try making Shape::draw non-pure by removing = 0 from its declaration.
3. Try changing shape (in main()) from a pointer to a stack-allocated variable.

# Assignment 2

Assume that a bank maintains two kinds of accounts for customers

1. one called as savings and
2. the other as current account.

The savings account provides compound interest and withdrawal facilities but no cheque book facility.

The current account provides cheque book facility but no interest. Current account holders should also maintain a minimum balance and if the balance falls below this level a service charge is imposed.

Create a class Account that stores customer name, account number and type of account.

From this derive the classes

1. CurrentAccount
2. SavingsAccount

to make them more specific to their requirements.

Include necessary member functions in order to achieve the following tasks:

1. Accept the deposit from a customer and update the balance.
2. Display the balance.
3. Compute and deposit interest.
4. Permit withdrawal and update the balance.
5. Check for the minimum balance, impose penalty, necessary and update the balance