**Trace the following:**

class base {

public:virtual void who() {cout << "Base\n";}};

class first\_d : public base {

public:void who() {cout << "First derivation\n";}};

class second\_d : public base {};

int main()

{

base base\_obj, \*p;

first\_d first\_obj;second\_d second\_obj;

p = &base\_obj;

p->who();

p = &first\_obj;

p->who();

p = &second\_obj;

p->who();

return 0;

}

-------------------------------------------------------------------------------------------------------------

class base {

public:virtual void who() {cout << "Base\n";}};

class first\_d : public base {

public:void who() {cout << "First derivation\n";}};

class second\_d : public first\_d {};

void main()

{

base base\_obj,\*p;

first\_d first\_obj;second\_d second\_obj;

p = &base\_obj;

p->who();p = &first\_obj;

p->who();

p = &second\_obj;

p->who();

}

----------------------------------------------------------------------------------

class base {

public:virtual void who() {cout << "Base\n";}};

class first\_d : public base {

public:void who() {cout << "First derivation\n";}};

class second\_d : public first\_d {

Public: void who() {cout << "second derivation\n";}};

void main()

{base base\_obj,\*p;

first\_d first\_obj;second\_d second\_obj;

p = &base\_obj;

p->who();

p = &first\_obj;

p->who();

p = &second\_obj;

p->who();

}

--------------------------------------------------------------------------------------

void myFunc();

int x = 6;

void main()

{

cout << "\n In main x is: " << x;

{

int x = 5;

cout << "\n In main x is: " << x;

myFunc();

cout << "\n Back in main, x is: " << x;

}

cout << "\n In main x is: " << x;

}

void myFunc()

{

int x = 8;

cout << "\n In myFunc, local x: " << x << endl;

{

cout << "\n In block in myFunc, x is: " << x;

int x = 9;

cout << "\n Very local x: " << x;

}

cout << "\n Out of block, in myFunc, x: " << x << endl;

}

----------------------------------------------------------------------------------------------------------------

Imagine a **tollbooth** at a bridge. Cars passing by the booth are expected to **pay a 50 Hlala** toll. Sometimes a car goes by without paying.

The tollbooth keeps track of the number of cars that have gone by, and of the total amount of money collected.

* Model this tollbooth with a **class** called **tollBooth**.
* The class contained **two** data items, the first one is from type **int** to hold the total number of cars.
* The second one is from type **double** to hold the total amount of money collected.
* A **constructor** initializes both of these variables to 0.
* A member function called **payingCar()** increments the car total and adds 0.50 to the cash total.
* A member function called **nopayCar**(), increments the car total but adds nothing to the cash total.
* A member function called **display()** displays the two totals.

Write a main () test this class.

Save the results in a file called “tollbooth.txt”

This program should allow the user to enter (1) count a paying car, and (0) to count a non-paying car.

Entering (-1) should cause the program to print out the total cars and total cash and then exit.

The run for the programme could be like this:

