Lab 4plus: ‘this’ pointer in C++

The ‘this’ pointer is passed as a hidden argument to all nonstatic member function calls and is available as a local variable within the body of all nonstatic functions. ‘this’ pointer is a constant pointer that holds the memory address of the current object. ‘this’ pointer is not available in static member functions as static member functions can be called without any object (with class name).
For a class X, the type of this pointer is ‘X* const’. Also, if a member function of X is declared as const, then the type of this pointer is ‘const X *const’

Following are the situations where ‘this’ pointer is used:

1) When local variable’s name is same as member’s name

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

/* local variable is same as a member's name */
class Test
{
private:
   int x;
public:
   void setX (int x)
   {
       // The 'this' pointer is used to retrieve the object's x
       // hidden by the local variable 'x'
       this->x = x;
   }
   void print() { cout << "x = " << x << endl; }
};

int main()
{
   Test obj;
   int x = 20;
   obj.setX(x);
   obj.print();
   return 0;
}
```

2) To return reference to the calling object

```
/* Reference to the calling object can be returned */
Test& Test::func ()
{
   // Some processing
   return *this;
}
```

When a reference to a local object is returned, the returned reference can be used to **chain function calls** on a single object.

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

class Test
{
private:
  int x;
  int y;
public:
  Test(int x = 0, int y = 0) { this->x = x; this->y = y; }
  Test &setX(int a) { x = a; return *this; }
  Test &setY(int b) { y = b; return *this; }
  void print() { cout << "x = " << x << " y = " << y << endl; }
};

int main()
{
  Test obj1(5, 5);

  // Chained function calls.  All calls modify the same object
  // as the same object is returned by reference
  obj1.setX(10).setY(20);

  obj1.print();
  return 0;
}
```

Exercise:

Predict the output of following programs. If there are compilation errors, then fix them.

Question 1

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

class Test
{
private:
	int x;
public:
	Test(int x = 0) { this->x = x; }

	void change(Test *t) { this = t; }

	void print() { cout << "x = " << x << endl; }
};

int main()
{
	Test obj(5);
	Test *ptr = new Test (10);
	obj.change(ptr);
	obj.print();
	return 0;
}
```

Question 2

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

class Test
{
private:
	int x;
	int y;
public:
	Test(int x = 0, int y = 0) { this->x = x; this->y = y; }
	static void fun1() { cout << "Inside fun1()"; }
	static void fun2() { cout << "Inside fun2()"; this->fun1(); }
};

int main()
{
	Test obj;
	obj.fun2();
	return 0;
}
```

Question 3

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

class Test
{
private:
    int x;
    int y;
public:
    Test (int x = 0, int y = 0) { this->x = x; this->y = y; }
    Test setX(int a) { x = a; return *this; }
    Test setY(int b) { y = b; return *this; }
    void print() { cout << "x = " << x << " y = " << y << endl; }
};

int main()
{
    Test obj1;
    obj1.setX(10).setY(20);
    obj1.print();
    return 0;
}
```

Question 4

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

class Test
{
private:
    int x;
    int y;
public:
    Test(int x = 0, int y = 0) { this->x = x; this->y = y; }
    void setX(int a) { x = a; }
    void setY(int b) { y = b; }
    void destroy()    { delete this; }
    void print() { cout << "x = " << x << " y = " << y << endl; }
};

int main()
{
    Test obj;
    obj.destroy();
    obj.print();
    return 0;
}
```