Aula Laboratório 3

1) Considere o seguinte troço de código em C#. Existem erros de compilação? Justifique.

```
struct MyInt {public int x;}
struct MyInt2 {public int x;}
class Test{
   static void m( ) {
     MyInt v1; v1.x=1;
     MyInt2 v2; v2.x=1;
     v2=v1;
 }
}
```

2) Considere o seguinte troço de código:

```
        struct Counter{
          int value;
          public override string ToString() {
            return value.ToString();
          }
          public void Increment() { value++;  }
        }
```

```
class Program  {
   static void Main() {
      Counter x = new Counter();
      object o = x;
      x.Increment(); Console.WriteLine(x.ToString());
      Console.WriteLine(o);
      ((Counter)o).Increment(); Console.WriteLine(o);
      Counter c = (Counter)x;
      c.Increment(); Console.WriteLine(c);
      Console.WriteLine(x);
   }
}
```

Diga, justificando, qual o *output* apresentado na consola após a execução do programa.

3) Dadas as definições `class A {public virtual void m(){...}}` e `class B:A{public virtual void m(){...}}`.
   a. Diga qual a relação entre os métodos `m` das duas classes.
   b. Considere a classe `C:B{public override void m(){...}}`.Qual a relação entre este método `m` e os outros?

4) Considere o seguinte troço de código.

```
struct Point {
int x, y;
public Point(int x, int y) {
this.x=x; this.y=y;
}
public void add( Point p, ref Point res) {
res = this;
res.x += p.x;
res.y += p.y;
}
public override String ToString() {
return String.Format("({0},{1})", x, y);
}
}
```

```
class Program {
static void Main(string[] args){
  Point p1 = new Point(1, 2),
  p2 = new Point(3, 4);
  p1.add(p2, ref p2);
  Console.WriteLine("p1={0}, p2={1}", p1,p2);
}
}
```

a) Indique e justifique a informação apresentada na consola resultante da execução da classe `Program`.
b) Indique e justifique as alterações no *output* no caso do tipo `Point` ser definido como tipo referência.

5) Considere a seguinte definição das classes `A, B, C e Printer`:

```
class A {
public virtual void m(){ Console.WriteLine("I'm A"); }
}
class B:A {
public override void m() { Console.WriteLine("I'm B"); }
}
class C:B {
public void m() { Console.WriteLine("I'm C"); }
}
class Printer {
public static void Print(C c) {
((A) c).m();
((B) c).m();
c.m();
}
}
```

a) Indique e justifique a informação apresentada na consola resultante da execução de `Printer.Print(new C());`
b) Considere a definição da interface

```
interface I { void m(); }
```

e as seguintes alterações às classes **`A`** e **`Printer`**:

```
class A: I {...}
class Printer {
public static void Print(I c) { … }
}
```

Em relação à alínea anterior justifique quais as diferenças verificadas na execução de `Printer.Print(new C());`

6) Considere o seguinte troço de código:

```
interface ICounter { void Increment(); }

struct Counter: ICounter {
  int value;
  public override string ToString() {
    return value.ToString();
  }
  public void Increment() { value++;  }
}
```

```
class Program  {
   static void Main() {
      Counter x = new Counter();
      object o = x;
      x.Increment(); Console.WriteLine(x.ToString());
      Console.WriteLine(o);
      ((ICounter)o).Increment(); Console.WriteLine(o);
      ICounter c = (ICounter)x;
      c.Increment(); Console.WriteLine(c);
      Console.WriteLine(x);
   }
}
```

Diga, justificando, qual o *output* apresentado na consola após a execução do programa.

7) Considere o seguinte troço de código.

```
public interface I {void Ave();}
public class A : I {
  void I.Ave() { Console.WriteLine("A -> I"); }
  public virtual void Ave() {
    Console.WriteLine("A");
  }
}
public class B : A, I {
  void I.Ave() { Console.WriteLine("B -> I"); }
  public new virtual void Ave() {
    Console.WriteLine("B");
  }
}
public class C : B {
  public override void Ave() {
    Console.WriteLine("C");
  }
}
```

```
static class Program{
  static void Main(){
      C c = new C();
      B b = c;
      A a = c;
      I i = c;

      c.Ave();
      b.Ave();
      a.Ave();
      i.Ave();
  }
}
```

a) Indique e justifique a informação apresentada na consola resultante da execução da classe `Program`.

b) Modificando apenas os atributos dos métodos, diga quais as alterações necessárias, de modo a que a execução das instruções `c.Ave(); b.Ave(); a.Ave();` apresentem na consola: C B A.

8)

```
interface I{
  void visit(Visitor v);
  void print();
}

struct A: I{
int n;
public void incN(int inc) {
    n += inc;
}
public void visit(Visitor v) {
     v.update(ref this);
}
public void print() {
Console.WriteLine("A =" + n);
}
}
```

```
interface Visitor{ void update(ref A a); }

class R1: Visitor{
  public void update(ref A a){a.incN(1);}
}

class R2: R1{
 public virtual void update(ref A a) { a.incN(2);}
}

class R3: R2, Visitor{
public override void update(ref A a){ a.incN(31); }
void Visitor.update(ref A a){a.incN(32);}
}

class Test{
static void Main(){
A a = new A(); a.incN(1); a.print();
a.visit(new R1()); a.print();
I i = a; ((A) i).incN(1); i.print();
i.visit(new R1()); i.print();
i.visit(new R2()); i.print();
i.visit(new R3()); i.print();
}
}
```

a) Diga e justifique qual o *output* resultante da execução do método `Main` da classe `Test`, identificando as operações de *boxing* e *unboxing*.

b) Considere a nova definição da interface `Visitor` –

```
interface Visitor{void update(ref I i);}
```

Ao recompilar a estrutura `A` para usar a nova interface `Visitor`, diga e justifique se é necessária, ou não, alguma alteração à implementação do método `visit`, para que compile com sucesso. Em caso afirmativo indique as modificações a efectuar.

c) Considerando que a assinatura de `update` é actualizada em cada uma das classes `R1`, `R2` e `R3` em conformidade com a nova definição da interface `I` e de acordo com as actualizações realizadas, ou não, à estrutura A no ponto anterior, indique e justifique as diferenças no output em relação ao cenário da alínea `a`.

9) Considere o seguinte troço de código.

```
interface Shape{
  void Print();
}
class Rectangle:Shape{
  public void Print(){Console.WriteLine("Rectangle");}
}
class Square:Rectangle {
  public void Print(){ Console.Write("Square");  }
}
```

```
static class Program{
  static void Main(){
    Rectangle r =
       new Square();
    r.Print();
    ((Shape) r).Print();
  }
}
```

a. Indique e justifique a informação apresentada na consola resultante da execução da classe `Program`.

b. Considere agora o seguinte troço de código. A informação apresentada na Consola é igual à alínea a? Justifique.

```
interface Shape{
  void Print();
}
class Rectangle:Shape{
  public void Print(){Console.WriteLine("Rectangle");}
}
class Square:Rectangle, Shape{
  public void Print(){ Console.Write("Square");  }
}
```

```
static class Program{
  static void Main(){
    Rectangle r =
       new Square();
    r.Print();
    ((Shape) r).Print();
  }
}
```

c. Indique e justifique as alterações às classes `Rectangle e Square` da alínea b para que o *output* seja `Square  Square`.

d. Indique e justifique as alterações às classes `Rectangle e Square` da alínea b para que o output seja `Rectangle  Rectangle`.