Nome: _____ Nr: _____

Cada pergunta certa contabiliza 2 valores, **errada 1 valor negativo**, sem resposta 0 valores.

Considere a definição da classe `Hashable` e uma possível extensão desta classe para a entidade `Student`.

```
public abstract class Hashable
{
  public int Seed(Type t) { …}
  public abstract int Hash(object target);
}
```

```
class HashableStudent : Hashable {
  public override int Hash(object target) {
    Student std = (Student) target;
    int hash = base.Seed(typeof(Student));
    hash = hash * 31 + ((object)std.Nr).GetHashCode();
    hash = hash * 31 + ((object)std.Name).GetHashCode();
    return hash;
  }
}
```

Implementou-se uma classe `HashableBuidler` com um método estático `Build(Type klass)` que retorna uma instância de uma classe derivada de `Hashable` para a entidade representada pelo parâmetro `klass`.
Ou seja, por **exemplo**, a chamada a `HashableBuilder.Build(typeof(Student))` retorna uma instância com uma implementação equivalente à de `HashableStudent`.

No âmbito da implementação de `HashableBuidler` considere nas questões seguintes que `klass` representa a tipo da entidade e `il` é o `ILGenerator` responsável por emitir o código do método `Hash`, que tem duas variáveis locais `local.0` e `local.1`, em que `local.0` é do tipo inteiro e `local.1` é do tipo da entidade representada por `klass`

1. A obtenção do `MethodInfo` para o método `Seed` de `Hashable` corresponde a:
   - ☐ `MethodInfo seed = typeof(Hashable).GetMethod("Seed", new Type[] { klass });`
   - ☐ `MethodInfo seed = typeof(Hashable).GetMethod("Seed", new Type[] { typeof(klass) });`
   - ✓ `MethodInfo seed = typeof(Hashable).GetMethod("Seed", new Type[] { typeof(Type) });`
   - ☐ `MethodInfo seed = typeof(Hashable).GetMethod("Seed", new Type[] { klass.GetType() });`

2. A emissão das instruções equivalentes a: `base.Seed(typeof(Student));` corresponde a: `il.Emit(OpCodes.Call, seed);`. Mas antes desta instrução:
   - ☐ são empilhados dois argumentos no *stack* em que o primeiro corresponde a: `il.Emit(OpCodes.Ldarg_1);`
   - ☐ é empilhado um argumentos no *stack* em que o primeiro corresponde a: `il.Emit(OpCodes.Ldarg_1);`
   - ✓ são empilhados dois argumentos no *stack* em que o primeiro corresponde a: `il.Emit(OpCodes.Ldarg_0);`
   - ☐ é empilhado um argumentos no *stack* em que o primeiro corresponde a: `il.Emit(OpCodes.Ldarg_0);`

3. A emissão das instruções equivalentes a: `Student std = (Student) target;` corresponde a:
   - ☐ `il.Emit(OpCodes.Ldarg_0); il.Emit(OpCodes.Castclass, klass); il.Emit(OpCodes.Stloc_1);`
   - ✓ `il.Emit(OpCodes.Ldarg_1); il.Emit(OpCodes.Castclass, klass); il.Emit(OpCodes.Stloc_1);`
   - ☐ `il.Emit(OpCodes.Ldarg_2); il.Emit(OpCodes.Castclass, klass); il.Emit(OpCodes.Stloc_1);`
   - ☐ `il.Emit(OpCodes.Ld_this); il.Emit(OpCodes.Castclass, klass); il.Emit(OpCodes.Stloc_1);`

4. Sendo `p` o representante de uma propriedade da entidade, então a emissão das instruções para obter o valor dessa propriedade corresponde a:
   - ☐ `il.Emit(OpCodes.Ldloc_1); il.Emit(OpCodes.Callvirt, p.GetMethod());`
   - ✓ `il.Emit(OpCodes.Ldloc_1); il.Emit(OpCodes.Callvirt, p.GetGetMethod());`
   - ☐ `il.Emit(OpCodes.Ldloc_1); il.Emit(p.GetGetMethod());`
   - ☐ `il.Emit(OpCodes.Ldloc_1); il.Emit(p.GetMethod());`

5. Sendo `f` o representante de um campo da entidade, como seria a emissão das instruções para obter o valor desse campo? (esta questão não tem penalização em caso de resposta errada)

   `il.Emit(OpCodes.Ldloc_1); il.Emit(OpCodes.Ldfld, f);`

   _____

6. A emissão das instruções equivalentes a: `hash * 31` corresponde a:
   - ☐ `il.Emit(OpCodes.Mul); il.Emit(OpCodes.Ldloc_0); il.Emit(OpCodes.Ldc_I4, 31);`
   - ☐ `il.Emit(OpCodes.Ldloc_0); il.Emit(OpCodes.Mul); il.Emit(OpCodes.Ldc_I4, 31);`
   - ✓ `il.Emit(OpCodes.Ldloc_0); il.Emit(OpCodes.Ldc_I4, 31); il.Emit(OpCodes.Mul);`
   - ☐ Nenhuma das opções.

7. Se `Nr` for uma propriedade de tipo `Int32` então a compilação da conversão para `(object)` resulta em:
   - ☐ `castclass  System.Object`
   - ✓ `box  System.Int32`
   - ☐ `unbox.any  System.Int32`
   - ☐ Nenhuma das opções.

8. Se `Name` for uma propriedade de tipo `String` então a compilação da conversão para `(object)` resulta em:
   - ✓ Nenhuma das opções.
   - ☐ `castclass  System.Object`
   - ☐ `box  System.String`
   - ☐ `unbox.any  System.String`

9. A emissão de instruções correspondente ao retorno do resultado do método `Hash` é:
   - ☐ `il.Emit(OpCodes.Ret, Ldloc_0);`
   - ✓ `il.Emit(OpCodes.Ldloc_0); il.Emit(OpCodes.Ret);`
   - ☐ `il.Emit(OpCodes.Ldloc_0);`
   - ☐ Nenhuma das opções.

10. O resultado da compilação da instrução c# `typeof(Student)` resulta em duas instruções IL onde a primeira é: `dtoken  Student` e a segunda é:
    - ☐ `call  Type::GetType(RuntimeTypeHandle)`
    - ✓ `call  Type::GetTypeFromHandle(RuntimeTypeHandle)`
    - ☐ `call  Type::GetType()`
    - ☐ `typeof`