

Za každý špatně zaškrtnutý řádek je 1 trestný bod.

U každé otázky může být více správných odpovědí. Správné odpovědi označte křížkem. Pokud se spletete, křížek zakroužkujte.

Pokud není explicitně uvedeno jinak, tak se všechny otázky týkají .NET Frameworku verze 4.0 a jazyka C# 4.0.

*Jaká bude hodnota v proměnné a po provedení kódu:*

```
int a = (24 | 4096) & (~(1 << 4));
```

- 4096  
 4100  
 4104  
 4112  
 4120

*Předpokládejme následující program:*

```
using System.Collections.Generic;
class Expression { }
class Plus : Expression {
    static Plus() {Prg.RegOp("+", 1); }
}
class Divide : Expression {
    static Divide() {Prg.RegOp("/", 2); }
}
class Prg {
    static Dictionary<string, int> ops =
        new Dictionary<string, int>();
    public static void RegOp(
        string op, int p) {
        ops.Add(op, p);
    }
    static void Main() {
        System.Console.Write(
            "{0},{1}", ops["+"], ops["/"]);
        new Plus();
        new Divide();
    }
}
```

*Jaký je jeho výstup?*

- Program se vůbec nepřeloží.  
 1,2  
 , (tj. pouze čárka)  
 Text výjimky (tj. program po spuštění skončí s chybou).  
 Nelze předem určit.

*Pro typ long určete velikost proměnných/datových položek tohoto typu:*

- 4 bajty  
 8 bajtů  
 32 bajtů  
 64 bajtů  
 2<sup>32</sup> bajtů  
 2<sup>64</sup> bajtů  
 Vždy závisí na cílové platformě (tj. zda je 32 nebo 64 bitová)

**Jméno a příjmení** .....

*Předpokládejme následující program:*

```
delegate int F();
class Prg { int a = 10;
    public F Adder(int x) {
        int i = x;
        return delegate {
            return a += i++; };
    }
    static void Main() {
        Prg p = new Prg();
        F f = p.Adder(5);
        p.Adder(10);
        f();
        System.Console.Write(f());
    }
}
```

*Jaký je jeho výstup?*

- Program se vůbec nepřeloží.  
 15  
 16  
 20  
 21  
 23  
 25  
 31

*V následujícím výčtu označte všechny hodnotové typy:*

- System.Object  
 object  
 decimal  
 System.Drawing.Color  
 System.Int64

*Ve vhodném kontextu předpokládejme úsek kódu A:*

```
var x = 1;
x = x.ToString();
```

*Ve vhodném kontextu předpokládejme úsek kódu B:*

```
System.Object x = 1;
x = x.ToString();
```

*Označte pravdivé výroky:*

- Úsek A nelze přeložit v žádném kontextu.  
 Úsek B nelze přeložit v žádném kontextu.  
 A i B lze přeložit a kód v úseku A JE ekvivalentní kódu v úseku B (A je zkratka za B).  
 A i B lze přeložit, ale kód v úseku A NENÍ ekvivalentní kódu v úseku B.

Označte všechny jazykové konstrukty, jejichž implementaci ve třídě může vynutit interface I1, který tato třída implementuje:

- operátor \*
- public metoda
- private metoda
- internal metoda
- implementaci jiného interfacu I2
- indexer (tj. operátor [])
- statická vlastnost (property)

**[Maximálně 2 trestné body (2 při prázdné odpovědi)]**

Předpokládejme následující zdrojový soubor:

```
class T { private int item;
    public int this[int offset] {
        get { return item + offset; }
    }
    public int this[long delta] {
        set { item =
            (int) delta + value; }
    }
    public int Item { get; set; }
}
```

Při pokusu o jeho přeložení vypíše překladač C# ve Visual Studiu 2008 i 2010 následující dvě chyby:

Line 2: The type 'T' already contains a definition for 'Item'

Line 5: The type 'T' already contains a definition for 'Item'

Popište (stručně, jasně, výstižně a čitelně) jaká chyba nastává a proč k ní dochází:

**[Maximálně 5 trestných bodů]**

Doplňte implementaci následující třídy tak, aby implementovala prioritní frontu pomocí dvojsměrně vázaného seznamu. Položky s větší hodnotou priority mají přednost před všemi položkami s hodnotou menší. Položky se stejnou prioritou se řadí v pořadí FIFO. Třidu napište tak, aby (a) za všech okolností zůstal její stav konzistentní (tj. mimo jiné, aby byla thread safe), (b) chybné použití indikovala vhodným způsobem (např. vyhozením vhodné výjimky).

```
public class Queue<T, P>
{
    .....Node {
        public Node prev, next;
        public T item;
        public P prio;
        public Node(T i, P p) {
            item = i; prio = p;
        }
    }

    public void Enqueue(T item, P prio) {
        Node n = new Node(item, prio);

    }
    public T Dequeue() {
        ...
    }
}
```