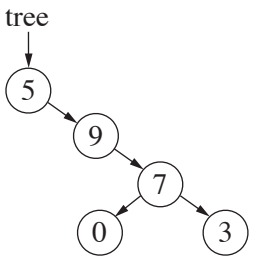
للّذين يحلُّون بلغة #C

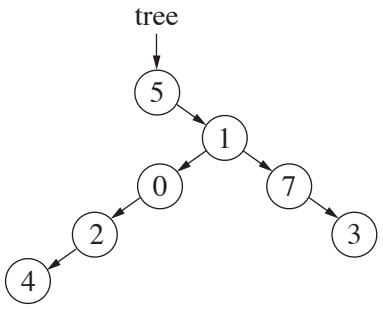
أ. معطاة العمليّة Foo التي أمامكم:

```
public static int Foo (BinNode<int> tree)
{
    if (tree.GetLeft() == null && tree.GetRight() == null)
        return 0;
    if (tree.GetLeft() == null)
        return Foo (tree.GetRight()) + 1;
    if (tree.GetRight() == null)
        return Foo (tree.GetLeft()) – 1;
    return Foo (tree.GetLeft()) + Foo (tree.GetRight());
}
```

(1) نفِّذوا متابعة للعمليّة Foo وللشجرة التي أمامكم، واكتبوا ماذا تُعيد العمليّة. يجب عرض المتابعة.



(2) نفِّذوا متابعة للعمليّة Foo وللشجرة التي أمامكم، واكتبوا ماذا تُعيد العمليّة. يجب عرض المتابعة.



(3) بالنسبة لشجرة فيها 6 عُقَد، ما هو أكبر عدد يمكن أن تُعيده العمليّة Foo ؟ اعرضوا الشجرة.

(4) بالنسبة لشجرة فيها 6 عُقَد، ما هو أصغر عدد يمكن أن تُعيده العمليّة Foo ؟ اعرضوا الشجرة.

(5) اكتبوا باختصار ماذا تُعيد العمليّة Foo بالنسبة لشجرة ما ليست فارغة.

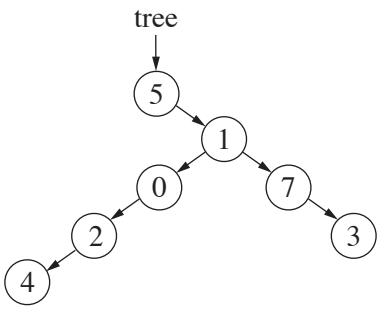
(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 9 /

ب. معطاة العمليّة Goo التي أمامكم:

```
public static bool Goo (BinNode<int> tree)
{
    if (tree == null)
        return true;
    if (Foo (tree) != 0)
        return false;
    return Goo (tree.GetLeft()) && Goo (tree.GetRight());
}
```

(1) نفِّذوا متابعة للعمليّة Goo وللشجرة التي أمامكم، واكتبوا ماذا تُعيد العمليّة. يجب عرض المتابعة.
في هذا البند، لا حاجة لعرض متابعة للعمليّة Foo .



(2) اعرضوا شجرة يوجد لها أكثر من 3 عُقَد بالنسبة لها تُعيد العمليّة Goo نتيجة عكسيّة للنتيجة التي نتجت في البند الفرعيّ "ب (1)".

/ يتبع في صفحة 10 /