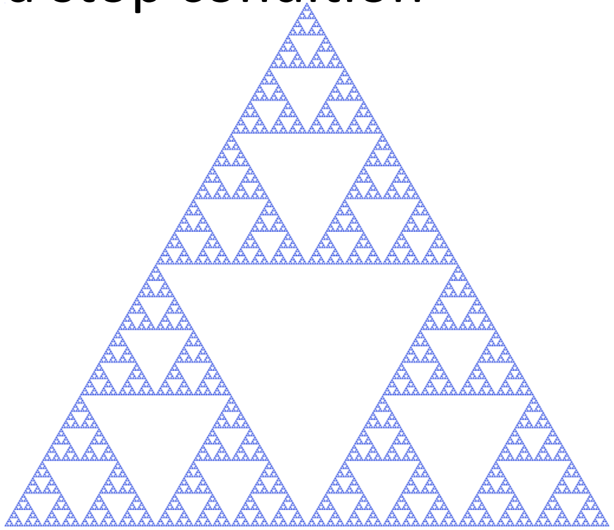
Introduction to Programming

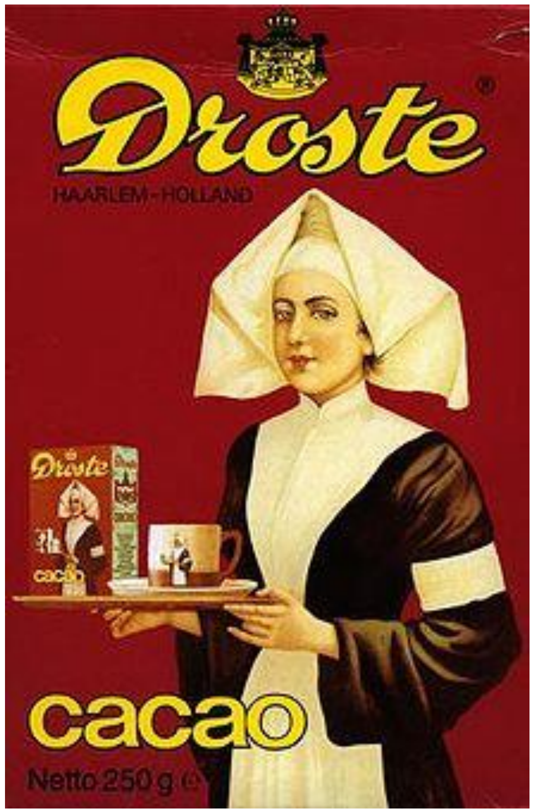
Lecture 12 – Recursion

Semester I 2013

Recursion

- A call to a function within the function
- Doing a similar action again and again
- Remember:
  - To make the recursive step
  - To create a stop condition





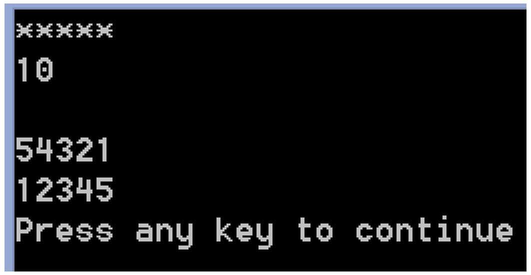
Recursion

- Example

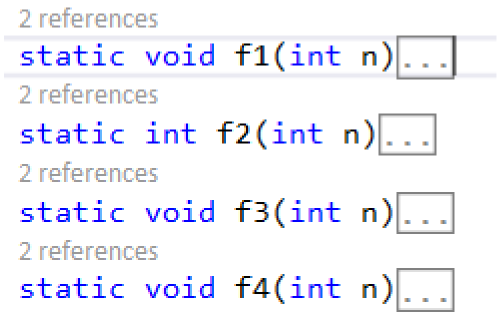
```
static int RecFactorial(int n)
{
    if (n == 1)
    {
        return 1;
    }
    return n * RecFactorial(n - 1);
}
```

Questions

```
f1(5);

Console.WriteLine();

Console.WriteLine( f2(1234) );

Console.WriteLine();

f3(5);

Console.WriteLine();

f4(5);
```



```
*****
10

54321
12345
Press any key to continue
```



```
2 references
static void f1(int n)...
2 references
static int f2(int n)...
2 references
static void f3(int n)...
2 references
static void f4(int n)...
```

Question

- יש לכתוב פונקציה רקורסיבית שקולטת שני מספרים שלמים X וY ומדפיסה את נוסחת הסכום בין הקטן לגדול ביניהם.
- לדוגמה:
- עבור X=5 ו-Y=2
- פלט:2+3+4+5

- עבור X=2 ו-Y=5
- פלט:2+3+4+5

Solution

Question

- יש לשנות את פונקציה כך שתדפיס את נוסחת הסכום בין הקטן לגדול מבין המספרים שנקלטו ותחשב את הערך.
- לדוגמה:
- עבור X=5 ו-Y=2
- פלט: 2+3+4+5
- 14

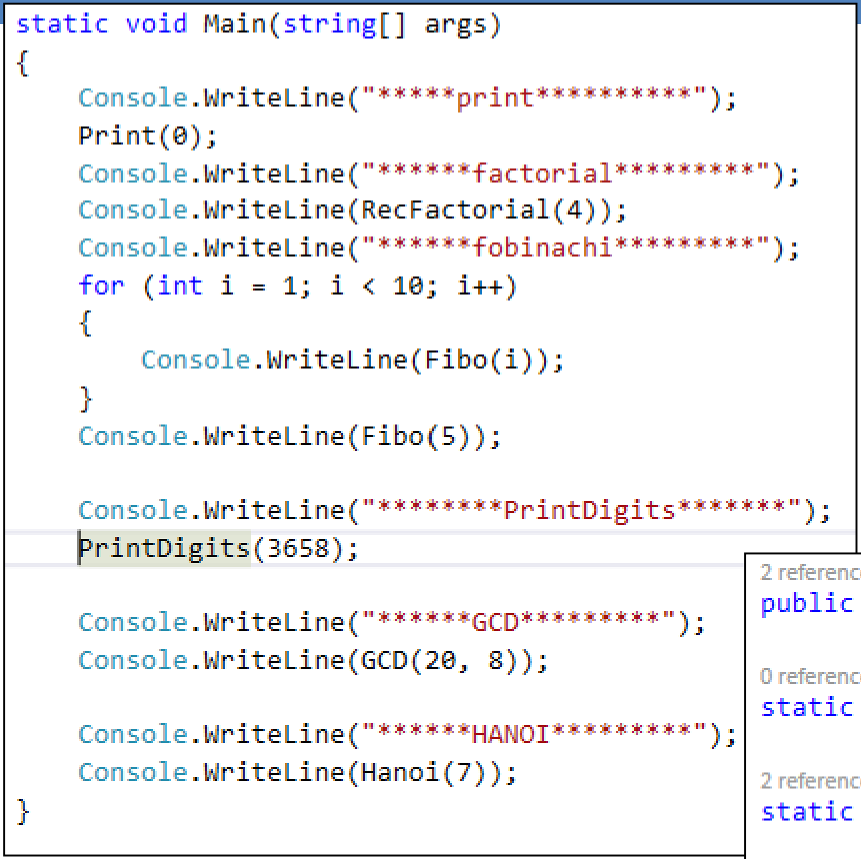
Solution

Fibonacci

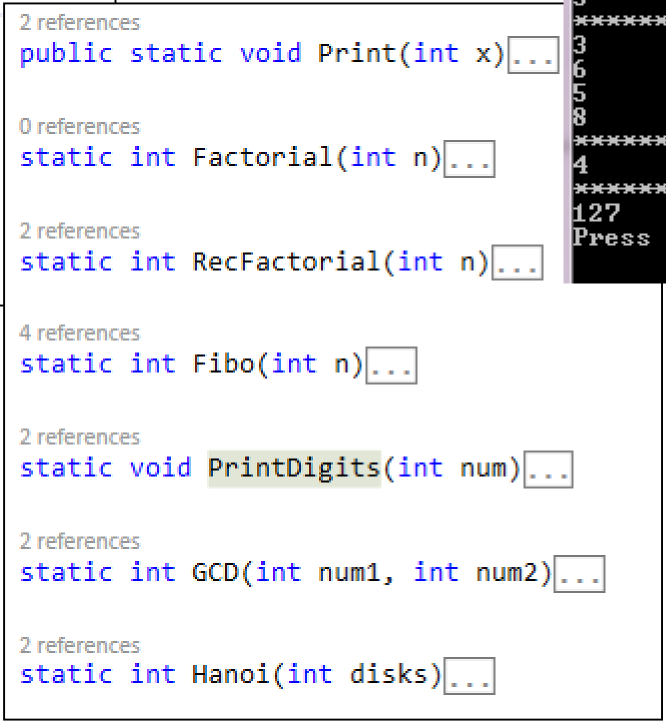
- In mathematics, the Fibonacci numbers or Fibonacci sequence are the numbers in the following integer sequence:
  - 1,1,2,3,5,8,13,21,34,55
  - FibonacciRec(7); → 13

Solution

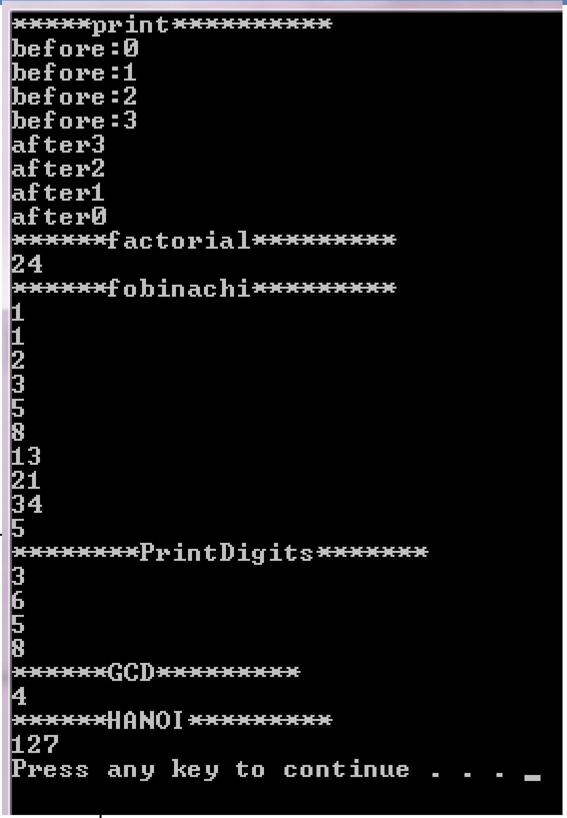
More q's



```
static void Main(string[] args)
{
    Console.WriteLine("*****print**********");
    Print(0);
    Console.WriteLine("******factorial*********");
    Console.WriteLine(RecFactorial(4));
    Console.WriteLine("******fobinachi*********");
    for (int i = 1; i < 10; i++)
    {
        Console.WriteLine(Fibo(i));
    }
    Console.WriteLine(Fibo(5));

    Console.WriteLine("********PrintDigits*******");
    PrintDigits(3658);

    Console.WriteLine("******GCD*********");
    Console.WriteLine(GCD(20, 8));

    Console.WriteLine("******HANOI*********");
    Console.WriteLine(Hanoi(7));
}
```

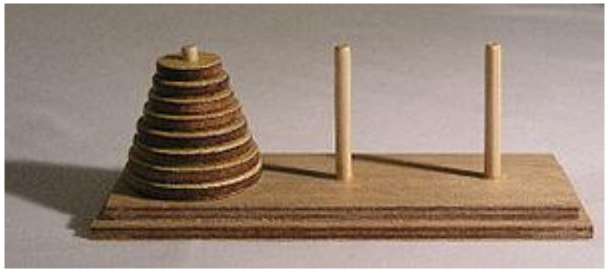


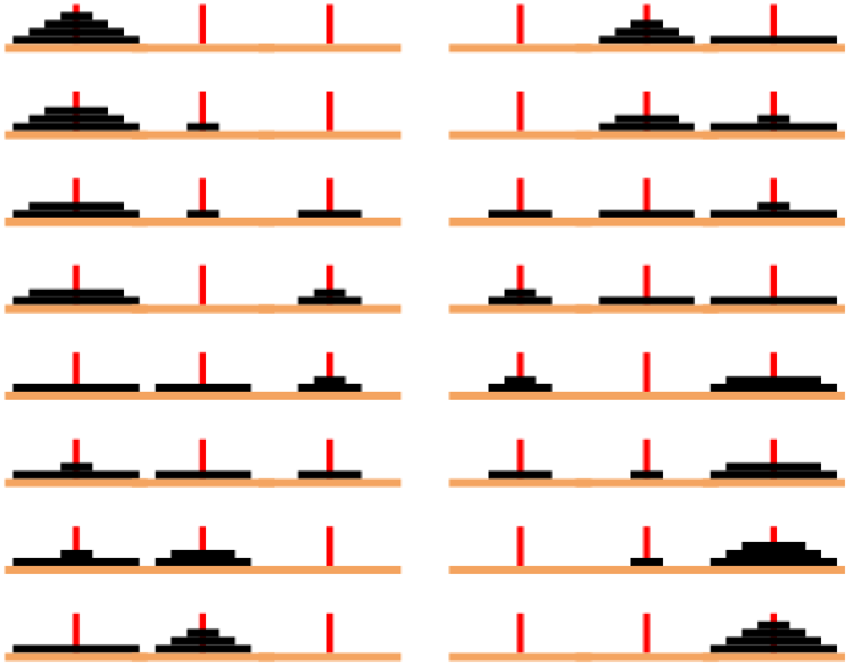
```
2 references
public static void Print(int x)...

0 references
static int Factorial(int n)...

2 references
static int RecFactorial(int n)...

4 references
static int Fibo(int n)...

2 references
static void PrintDigits(int num)...

2 references
static int GCD(int num1, int num2)...

2 references
static int Hanoi(int disks)...
```



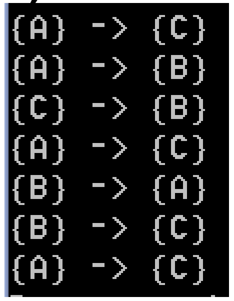
```
*****print**********
before:0
before:1
before:2
before:3
after3
after2
after1
after0
******factorial*********
24
******fobinachi*********
1
1
2
3
5
8
13
21
34
5
********PrintDigits*******
3
6
5
8
******GCD*********
4
******HANOI*********
127
Press any key to continue . . . _
```

Tower of Hanoi





- void HanoiTowers(int numOfDisks, char source, char dest, char temp)
- HanoiTowers(3, 'A', 'C', 'B');

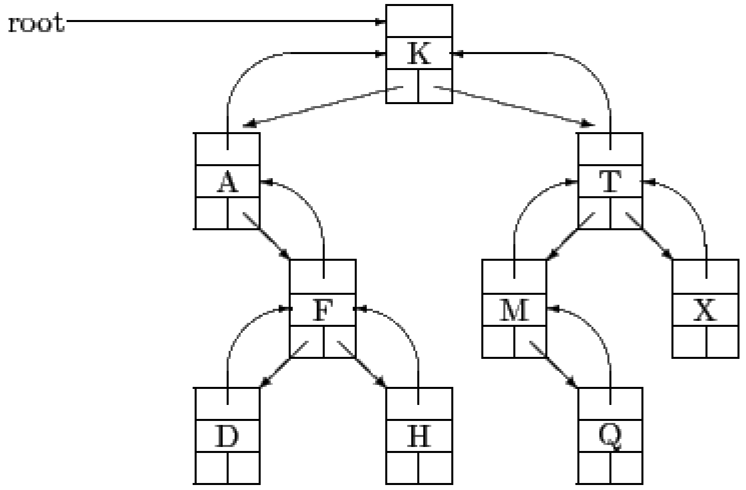


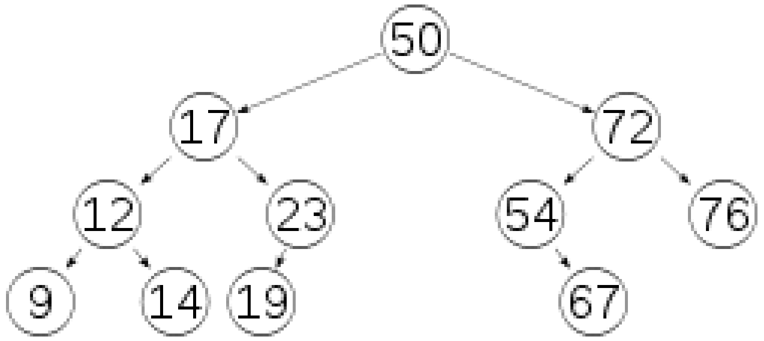
```
{A} -> {C}
{A} -> {B}
{C} -> {B}
{A} -> {C}
{B} -> {A}
{B} -> {C}
{A} -> {C}
```

Solution

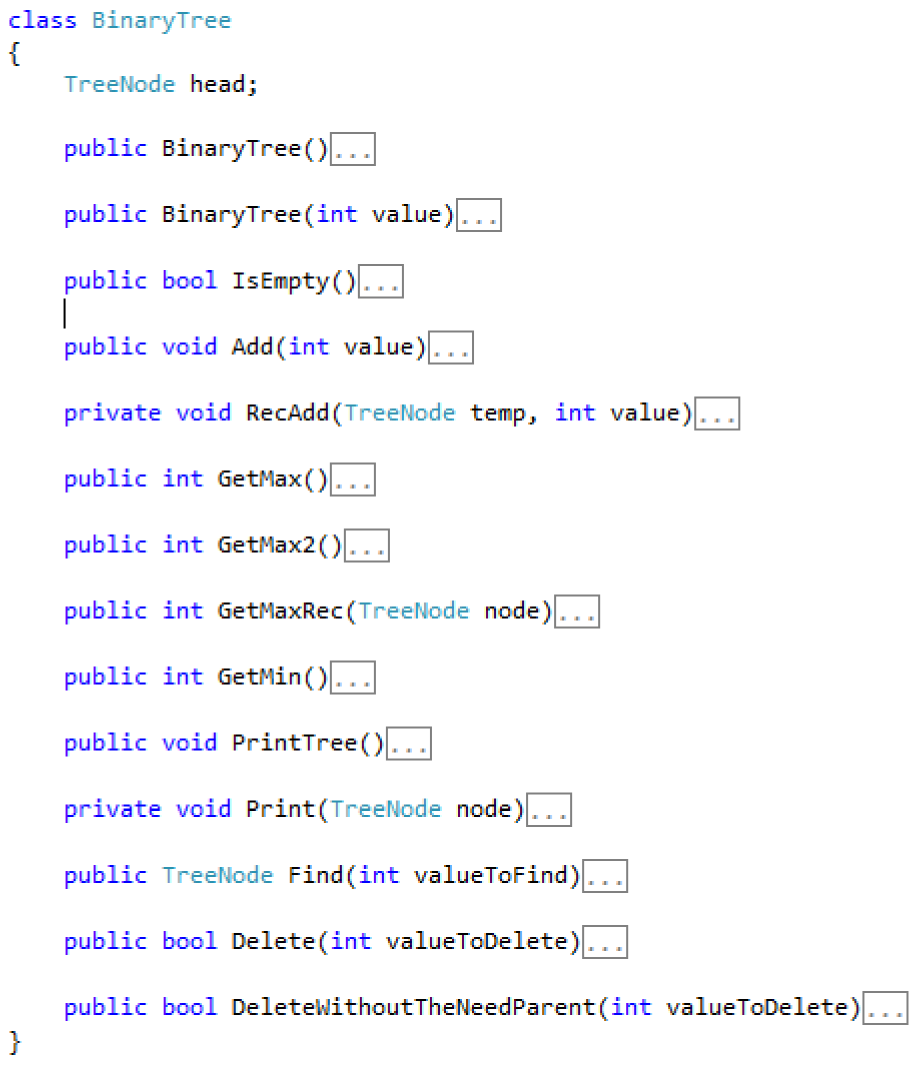
Binary Tree

- Every TreeNode has a value a Left, Right (and in our case a Parent) reference to a TreeNode

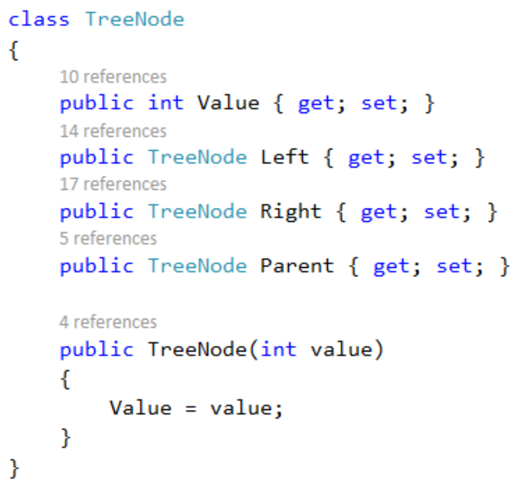




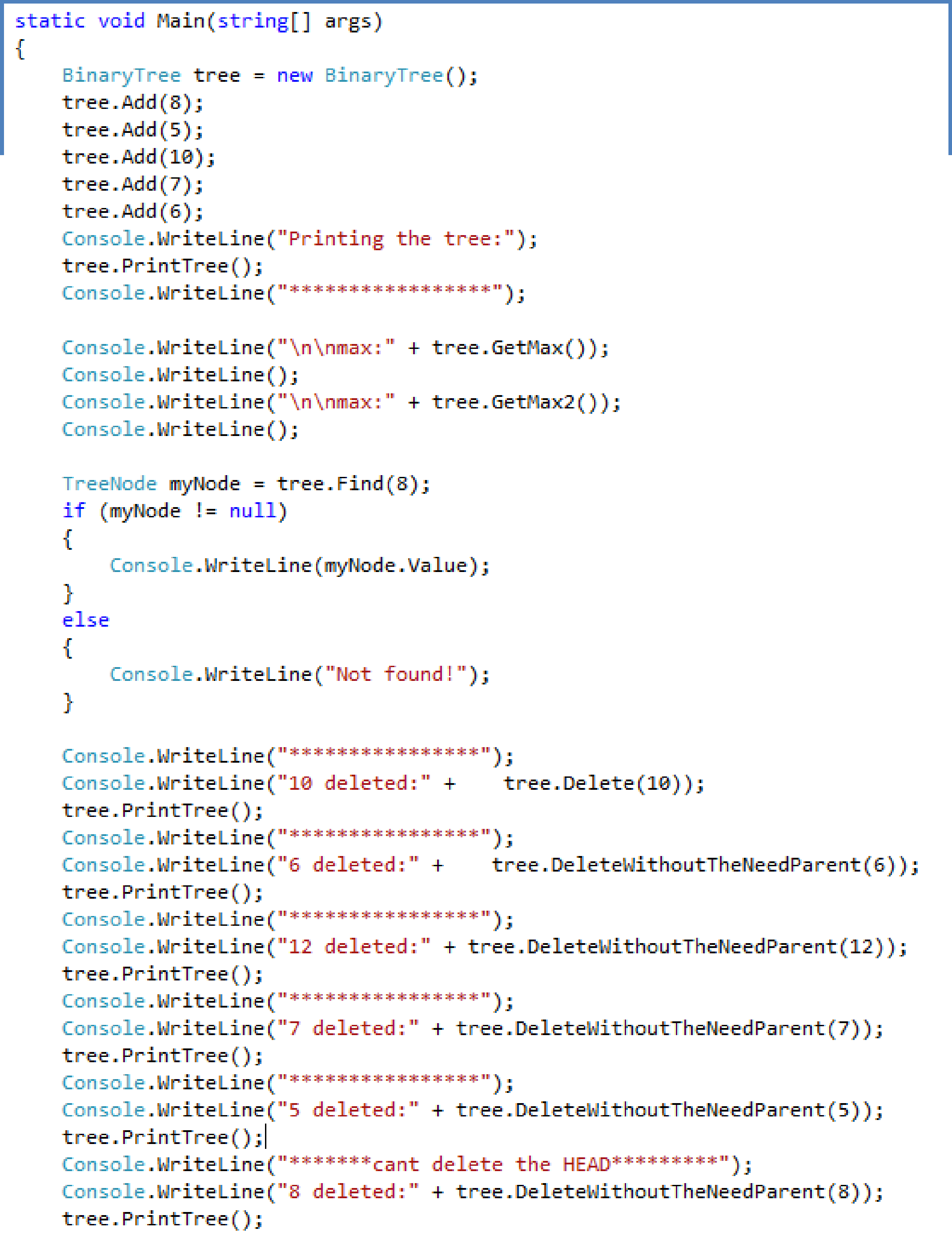
Binary tree



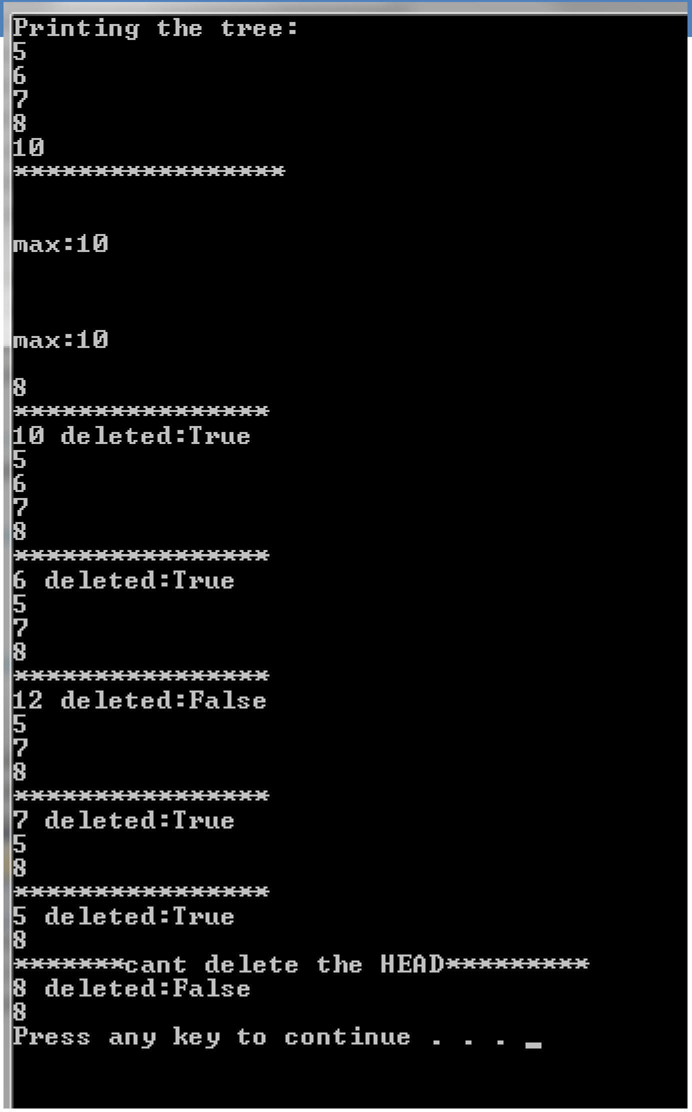
```
class BinaryTree
{
    TreeNode head;

    public BinaryTree()...

    public BinaryTree(int value)...

    public bool IsEmpty()...

    public void Add(int value)...

    private void RecAdd(TreeNode temp, int value)...

    public int GetMax()...

    public int GetMax2()...

    public int GetMaxRec(TreeNode node)...

    public int GetMin()...

    public void PrintTree()...

    private void Print(TreeNode node)...

    public TreeNode Find(int valueToFind)...

    public bool Delete(int valueToDelete)...

    public bool DeleteWithoutTheNeedParent(int valueToDelete)...
}
```



```
class TreeNode
{
    10 references
    public int Value { get; set; }
    14 references
    public TreeNode Left { get; set; }
    17 references
    public TreeNode Right { get; set; }
    5 references
    public TreeNode Parent { get; set; }

    4 references
    public TreeNode(int value)
    {
        Value = value;
    }
}
```



```
static void Main(string[] args)
{
    BinaryTree tree = new BinaryTree();
    tree.Add(8);
    tree.Add(5);
    tree.Add(10);
    tree.Add(7);
    tree.Add(6);
    Console.WriteLine("Printing the tree:");
    tree.PrintTree();
    Console.WriteLine("*****************");

    Console.WriteLine("\n\nmax:" + tree.GetMax());
    Console.WriteLine();
    Console.WriteLine("\n\nmax:" + tree.GetMax2());
    Console.WriteLine();

    TreeNode myNode = tree.Find(8);
    if (myNode != null)
    {
        Console.WriteLine(myNode.Value);
    }
    else
    {
        Console.WriteLine("Not found!");
    }

    Console.WriteLine("****************");
    Console.WriteLine("10 deleted:" +    tree.Delete(10));
    tree.PrintTree();
    Console.WriteLine("****************");
    Console.WriteLine("6 deleted:" +    tree.DeleteWithoutTheNeedParent(6));
    tree.PrintTree();
    Console.WriteLine("****************");
    Console.WriteLine("12 deleted:" + tree.DeleteWithoutTheNeedParent(12));
    tree.PrintTree();
    Console.WriteLine("****************");
    Console.WriteLine("7 deleted:" + tree.DeleteWithoutTheNeedParent(7));
    tree.PrintTree();
    Console.WriteLine("****************");
    Console.WriteLine("5 deleted:" + tree.DeleteWithoutTheNeedParent(5));
    tree.PrintTree();
    Console.WriteLine("*******cant delete the HEAD*********");
    Console.WriteLine("8 deleted:" + tree.DeleteWithoutTheNeedParent(8));
    tree.PrintTree();
```



```
Printing the tree:
5
6
7
8
10
*****************


max:10



max:10

8
****************
10 deleted:True
5
6
7
8
****************
6 deleted:True
5
7
8
****************
12 deleted:False
5
7
8
****************
7 deleted:True
5
8
****************
5 deleted:True
8
*******cant delete the HEAD*********
8 deleted:False
8
Press any key to continue . . . _
```

Red Black Tree

- Try to maintain a balanced tree all the time…RB-Tree