```
public class MyList<T>
    {
        private T item;
        private MyList<T> next;

        public MyList(T item)
        {
            this.item = item;
        }

        public T GetItem(int index)
        {
            if (index == 0)
                return item;
            else
                return next.GetItem(index - 1);
        }

        public void Add(T item)
        {
            if (next == null)
                next = new MyList<T>(item);
            else
                next.Add(item);
        }
    }

try {
    int c = list.GetItem(42);
}
catch (NullReferenceException e) {
    Console.WriteLine("index out of bounds");
}
catch (Exception e) {
    Console.WriteLine("Error! ");
}

class Tree {
    private int item;
    private Tree left = null;
    private Tree right = null;
    public Tree(int item) {
        this.item = item;
    }
public void Add(int item) {
    if (item != this.item) {
        if (item < this.item)
            if (left == null)
                left = new Tree(item);
            else
                left.Add(item);
        else
            if (right == null)
                right = new Tree(item);
            else
                right.Add(item);
    }
}
public bool Search(int item) {
    if (item == this.item)
        return true;
    if (item < this.item)
        if (left == null)
            return false;
        else
            return left.Search(item);
    else
        if (right == null)
```

```
            return false;
        else
            return right.Search(item);
    }
}
}
```

Faculteit

```
 public int fac(int n) {
     if (n == 1)
         return 1;
     else
         return n * fac(n-1);
 }
```

Fibonacci

```
 public int fib(int n) {
     if (n == 0)
         return 0;
     else if (n == 1)
         return 1;
     else
         return fib(n-1) + fib(n-2);
 }
```

Insertion Sort

```
 private List<int> Insert(int element, List<int> list) {
     List<int> result = new List<int>();
     if (list.Count == 0) {
         result.Add(element);
     } else {
         if (element < list[0]) {
             result.Add(element);
             result.AddRange(list);
         } else {
             result.Add(list[0]);
             list.RemoveAt(0);
             result.AddRange(Insert(element, list));
         }
     }
     return result;
 }

 public string reverseString(string s) {
     if (s.isEmpty())
         return ””;

     return reverseString(s.substr(1)) + s[0];
 }

public double PiDecimalen(int n) {
    double deler = (n*2.0)*((n*2)+1)*((n*2)+2);
    if (n == 1)
        return 4.0/deler;
    else
        if (n % 2 == 0)
            return PiDecimalen(n-1) – (4.0 / deler);
        else
            return PiDecimalen(n-1) + (4.0 / deler);
}
```