Xxx

|  |  |  |
| --- | --- | --- |
| 1.1 | [TestFixture]  public class test\_RPNCalculator  {      [Test]      public void Push\_on\_empty\_stack()      {          var sut = new RPNCalculator();      }  } | public class RPNCalculator  {  } |
| 1.2 | [Test]  public void Push\_on\_empty\_stack()  {      var sut = new RPNCalculator();        var result = sut.Push(1);  } | public class RPNCalculator  {      public Tuple<IEnumerable<int>, int>  Push(int number)      {          throw new **NotImplementedException**();      }  } |
| 1.3 | [Test]  public void Push\_on\_empty\_stack()  {      var sut = new RPNCalculator();        var result = sut.Push(1);        Assert.AreEqual(new[]{1}, result.Item1);      Assert.AreEqual(1, result.Item2);  } | public Tuple<IEnumerable<int>, int>  Push(int number)  {      return new Tuple<IEnumerable<int>, int>(new[]{number}, number);  } |
| 2.1 | [Test]  public void Push\_number\_on\_non\_empty\_stack()  {      var sut = new RPNCalculator(initialStack);  } | public class RPNCalculator  {      private readonly Stack<int> \_stack;        public RPNCalculator() : this(new Stack<int>()) {}      public RPNCalculator(Stack<int> stack)      {          \_stack = stack;      } |
| 2.2 | [Test]  public void Push\_number\_on\_non\_empty\_stack()  {      var initialStack = new Stack<int>();      initialStack.Push(1);      var sut = new RPNCalculator(initialStack);      var result = sut.Push(2);        Assert.AreEqual(new[] { 1, 2 }, result.Item1);      Assert.AreEqual(2, result.Item2);  } | public Tuple<IEnumerable<int>, int>  Push(int number)  {      \_stack.Push(number);      return new Tuple<IEnumerable<int>, int>(\_stack.Reverse(), number);  } |
|  |  |  |

yyy