Exercise 1:

Your first exercise is to use the Russian Peasant's Algorithm to multiply the following two integers by hand:

68 x 139

139	68	139
278	34	278
556	***17***	***556***
1112	8	1112
2224	4	2224
4448	2	4448
8896	***1***	***8896***

556 + 8896 = 9452 (Correct Answer)

Exercise 2 Option B

```
import
java.math.BigInteger;
                        import java.util.Scanner;
                        import java.util.concurrent.TimeUnit;

                        public class mainRussianPeasantAlgorithm {
                               public static void main(String args[]) {

                                       //scanner to accept user data
                                       Scanner s = new Scanner(System.in);

                                       //text to tell user to input the first number
                                       System.out.println("Please enter the first
                        integer");

                                       //accept values for bigIntegers
                                       BigInteger number1 = s.nextBigInteger();
                                       System.out.println("Please enter the second
                        integer");
                                       BigInteger number2 = s.nextBigInteger();


                                       //some variables to allow for russian peasant
                        algorithm in BigInteger
                                       BigInteger result = new BigInteger("0");
                                       BigInteger mod2 = new BigInteger("2");
                                       BigInteger mod2answer = number1.mod(mod2);
                                       BigInteger one = new BigInteger("1");
```

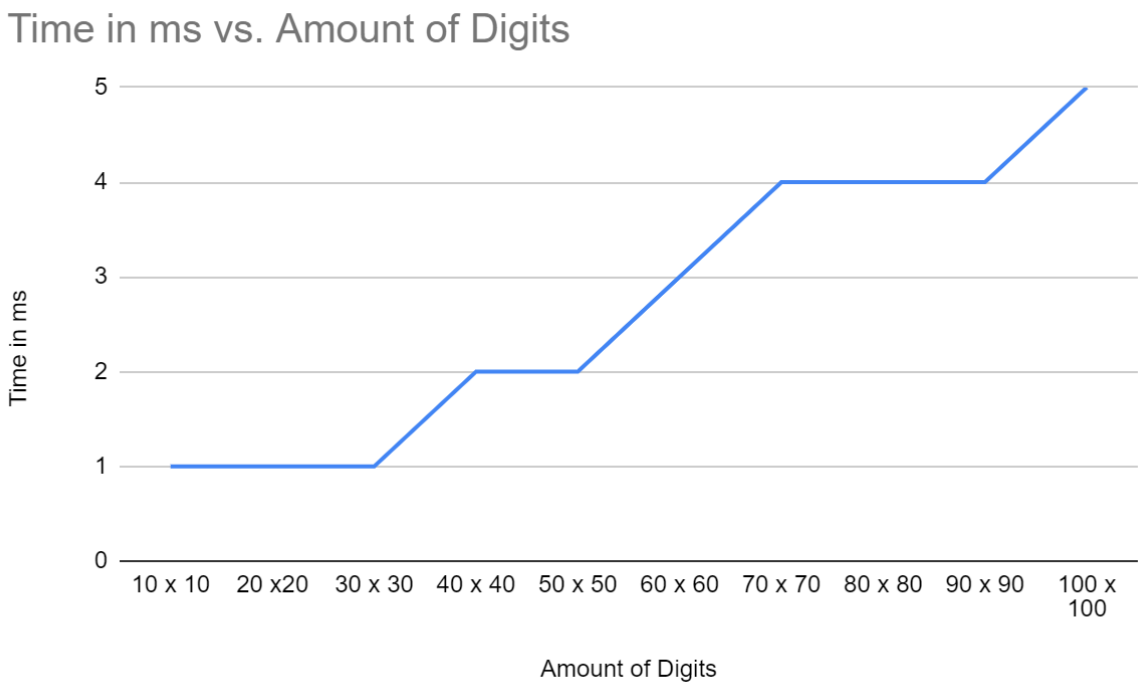
```
                //measuring time
                long startTime = System.currentTimeMillis();

                //first part of the russian peasant algorithm
                //if the first number is odd, add the second
number to the result
                if(number1.mod(mod2).equals(one)) {
                        result =result.add(number2);
                }
                //while the first number != one divide number1 by
two
                //and multiply number2 by two
                while(!(number1.equals(one))) {
                        number1 = number1.divide(mod2);
                        number2= number2.multiply(mod2);
                        if(number1.mod(mod2).equals(one)) {
                                //if the first number is odd, add
the second number to the result
                                result =result.add(number2);
                        }
                }
                System.out.println(result);

        //the code being measured ends
        long endTime = System.currentTimeMillis();
        long timeElapsed = endTime - startTime;
        System.out.println("Execution time in milliseconds: " +
timeElapsed);

       }
}
```

Exercise 3



I think the space complexity is constant O(1) because you only need to store 2 BigIntegers and this doesn't change depending on how big the 2 BigIntegers are.

I think the time complexity is O(logn) as the time increases logarithmically in comparison to the input size which increases at a steady rate.