Zadanie 1 (4p)
Zrealizować w postaci programu np. w Pythonie jakikolwiek algorytm generacji liczb pseudo-losowych bezpieczny kryptograficznie (np.algorytm BBS).

```
   ArrayList<BigInteger> generatorBBC ( int len ) {
        return generatorBBC( 11L,23L,3L,len );
    }

    ArrayList<BigInteger> generatorBBC ( Long p, Long q, Long s, int len ){

        BigInteger p1=BigInteger.valueOf( p ); // p=11
        BigInteger q1= BigInteger.valueOf( q ); // q=23
        BigInteger ziarno = BigInteger.valueOf( s ); // s=3
        BigInteger n = p1.multiply( q1 );
        ArrayList<BigInteger> myList = new ArrayList<>();

        for ( int i=0;i<len;i++ ){
            ziarno = (( ziarno.multiply( ziarno )).mod( n ));
            if ( myList.contains( ziarno ) ) {
                System.out.println("petla! i="+i );
                return myList;
            }
            myList.add( ziarno );
        }
        return  myList;
    }

// [9, 81, 236, 36, 31, 202, 71, 234, 108, 26, 170, 58, 75, 59, 192]
```

Zadanie 2 (2p)
Wykorzystując bibliotekę Open SSL napisać skrypt obliczający funkcję skrótu z zadanego pliku. Podać przykłady obliczeń dla różnych plików tekstowych stanowiących dane wejściowe. Zastosować kolejno następujące funkcje skrótu dostępne w Open SSL:
MD5, SHA, SHA-256, SHA-512

Zadanie 3 (1p)
Wykorzystując bibliotekę Open SSL napisać skrypt obliczający kluczowaną funkcję skrótu MAC np. wykorzystując funkcję skrótu MD5. Podać przykłady obliczeń dla różnych plików i kluczy.

Zadanie 4 (1p)
Wykorzystując bibliotekę Open SSL napisać skrypt szyfrujący szyfrem RSA i skrypt deszyfrujący szyfrem RSA. Podać przykłady szyfrowania dla różnych plików i kluczy.

Zadanie 5 (1p)
Jakie jest prawopodobieństwo tego, że losowo wygenerowana nieparzysta liczba złożona n zostanie w teście Millera-Rabina uznana za liczbę pierwszą, przy założeniu, że dokonujemy t losowań podstawy a ?

parametr t określa prawdopodobieństwo popełnienia pomyłki przy raz wybranym a prawdopodobieństwo pomyłki (wskazania liczby pierwszej nie będącej pierwszą) wynosi nie więcej niż $1/4^t$. Jeśli a wybieramy t-krotnie - wówczas to prawdopodobieństwo spada do $1/t*4^t$

Czy liczba pierwsza może być w teście Millera -Rabina uznana za liczbę złożoną? - nie

Ile wykonać niezależnych doświadczeń-losowań podstawy a w algorytmie badania pierwszości Millera – Rabina, by z prawdopodobieństwem większym równym od mieć pewność, że testowana liczba n jest pierwsza. - Nieskończenie wiele lub użyć innego algorytmu.

Zadanie 6 (8p)
Napisać w Pythonie program ustanawiający szyfr RSA
tzn. dobierający parametry szyfru RSA oraz szyfrujący i deszyfrujący.

```
   private final Integer e=65537;
    private final BigInteger d;
    private final BigInteger n;




    public  RSA (){
        BigInteger p =  ( BigInteger.TWO.pow( 2000 ).multiply( BigInteger.valueOf(new Random().nextInt(0,1000))));
                    p = p.nextProbablePrime();
        BigInteger q =  ( BigInteger.TWO.pow( 2000 ).multiply( BigInteger.valueOf(new Random().nextInt(0,1000))));
                    q = q.nextProbablePrime();
    BigInteger CDofN = ( p.subtract(BigInteger.ONE).multiply( q.subtract(BigInteger.ONE)));
                    n = p.multiply(q);
        BigInteger be = BigInteger.valueOf( e );
                    d = be.modInverse( CDofN );

        System.out.println("p:"+p);
        System.out.println("q:"+q);
        System.out.println("CDofN: "+CDofN);
        System.out.println("n: "+n);
        System.out.println("d:"+d);
        System.out.println("e:"+e);
        System.out.println( d.multiply(be).mod( n ).toString());

    }

    public BigInteger encrypt ( String s ) {
        BigInteger svalue=BigInteger.ZERO;
        char[] chars = s.toCharArray();
        for ( int i=0;i<chars.length;i++ ){
            svalue=svalue.multiply( BigInteger.valueOf(256L)).add( BigInteger.valueOf( chars[i] ));
        }
        System.out.println( "SVal:" + svalue );
        BigInteger C = svalue.modPow( BigInteger.valueOf(e), n ); //  .pow( e ).mod(n);
        return C;
    }

    public String decrypt ( BigInteger C ) {
        BigInteger m = C.modPow( d, n );
        StringBuffer out=new StringBuffer();

        System.out.println( "SVal:" + m );
        byte[] bary = m.toByteArray();
        for ( int i=0;i<bary.length ; i++ ){
          out.append( (char) (128+bary[i]) );
        }
        return out.toString();
    }
```

SVal IN :**110873100365438973320959359753048386337**

Encrypt: 92572170894503471953910204815260769880024013392199346992345899106132286965066653943234674576486638785966971
86382485599895174770139024048299605243745517287565995802049712445039958980800865857336284053731201160369430495967357
44545793454677191442544798381800336480771406962907211329435783067262593785197442287895911652234776140592793002036572
04258670625806667974781211216604121696217537776726375287060380851746759410257442450804166380641487731379785651565974
24582321384040476062176274060076782580637453604885046019185114682683237006131531812308284257488913928508392651172934
74024386091819418094763493434032375710029521762320922034795967958318512498582605102579283921681164587740121948456580
64947792502169106625485309957471518146948411517300385073848048735130248505427787792120976006115270235978905932999143
25130369482604394196216654678366051951678127641718093789830487300825682462156399996061814395245124668659139367848744
32496623767666582552084613207136265133697847545201686417202114911550834001981590342429741326684484844349768163175679
71306347839000171603810930479485549224981656989393769667985812608774758834714031858293002923997944812903109871537230
31120933758620393412512928247567438653971435580282224765

SVal OUT:**110873100365438973320959359753048386337**

Zadanie 7 (8p)
Napisać w Pythonie program implementujący test pierwszości Millera-Rabina.

```
  public boolean Miller_Rabbin_isPrime ( Integer s , BigInteger r , Long t ){
        BigInteger n = BigInteger.TWO.pow( s ).multiply( r ).add( BigInteger.ONE );
        return Miller_Rabbin_isPrime( BigInteger.valueOf(s), r , t , n );
    }


    public boolean Miller_Rabbin_isPrime ( BigInteger s , BigInteger r , Long t , BigInteger n ){
        for ( Long i=0L; i<t ; i++){
            BigInteger a;
            BigInteger j;
            BigInteger y;
            do {
                a = generatorBBC(10).get(0).add(BigInteger.TWO);
            } while ( a.compareTo( n ) < 0 );
            BigInteger m=( n.subtract( BigInteger.ONE ).divide( BigInteger.TWO ) );
            j=s;
               while( j.toString().indexOf("-")>0 ){
                   y = a.modPow( m, n ).mod( n );
                   if ( y.mod(n).compareTo( BigInteger.ONE )!=1L && y.compareTo(  ( n.subtract(BigInteger.ONE)).mod(n)  )!=0 ){
                       System.out.println("n jest liczba złożoną");
                       return false;
                   }
                   if ( y.compareTo(  ( n.subtract(BigInteger.ONE)).mod(n)  )==0  ){
                       j=BigInteger.ZERO;
                   } else {
                       j=j.subtract(BigInteger.ONE);
                       m=m.divide(BigInteger.TWO);
                   }
               }
        }
        System.out.println("n jest liczba pierwszą");
        return true;
    }
```

```
n jest liczba pierwszą: 2097153
n jest liczba złożoną: 1917273370
```