```java

public class LargestPrimeFactor {

      public static void main(String args[])
        {
            //the number to factor
            long numToFactor = 600851475143L;

            //for loop to iterate through the range
            for(int i=2;i<numToFactor ; i++)
            {
                //while loop to determine if factors are divisible or not
                while(numToFactor %i == 0)
                {
                    numToFactor =numToFactor /i;
                }
            }
            if(numToFactor >2)
                System.out.println(numToFactor );

        }

    }
public class tenThousandOnePrimeNumber {


    public static boolean isPrime(long num)
    {
        //finds the max number of divisors for a prime number
        int numSkipDivisor = 5;
        int max = (int)Math.floor((int)Math.sqrt(num));
        if (num <= 1)
        {
            return false;

        }

        //uses the formula 6K±1 = to find prime numbers. Loop below skips
        //ever six numbers when finding the divisors of the number.

        while (numSkipDivisor <= max)
        {
            if (num % numSkipDivisor == 0)
            {
                return false;

            }
            if (num % (numSkipDivisor + 2) == 0)
            {
                return false;

            }
            numSkipDivisor+= 6;
        }
        return true;

```

```java
    }

    public static void main(String[] args) {
            int primeCount = 2;
            long valueToIterate= 0;

            while (primeCount < 10001) {

            valueToIterate += 6;

            //accounts for the + 1 case in the formula 6K±1
            if (isPrime(valueToIterate + 1)) {
                primeCount++;
            }
            //accounts for the - 1 case in the formula 6K±1
            if (isPrime(valueToIterate - 1)) {
                primeCount++;
            }
            }
            // Add one to the prime number to get it into the form 6k + 1
            System.out.print(valueToIterate+1);
        }
}


public class SpecialPythagoreanTriplet {

    public static void main(String[] args) {

        //for loop to parse through the first 1000 terms (could make this more efficient by
        //only looping through the relative max value for a instead of the absolute sum?)
        for (int a = 3; a <= 1000; a++)
            {
                //loop to iterate through all the possible values of b, starting with the
                //immediate value after a.
                for (int b = a + 1; b < 1000; b++)
                {

                    //calculate the sum of a and b squared
                    double sumSquared = Math.pow(a, 2) + Math.pow(b, 2);
                    double c = Math.pow(sumSquared, 0.5);

                    //print the final result if the condition is met
                    if (a + b + c == 1000)
                    {
                        System.out.println(a * b * c);
                    }
                }
            }
    }

}


public class EvenFibonacci {

    static int sum = 0;
```

```
115
116     public static void fibonacci (int FnMinus1, int Fn)
117     {
118             int fibonacciSequence = Fn;
119             if (fibonacciSequence <= 4000000)
120             {
121                 //checks if the nth fibonacci number is even and adds
122                 //it to the sum if it is
123                 if (fibonacciSequence % 2 == 0)
124                 {
125                     sum += fibonacciSequence;
126                 }
127                 //recursive aspect to find the nth term of fibonacci sequence
128                 fibonacci(fibonacciSequence, FnMinus1 + fibonacciSequence);
129             }
130
131             else
132             {
133                 System.out.println(sum);
134             }
135         }
136
137     public static void main (String[] args)
138     {
139         //value to kick start the sequence
140         int seq = 2;
141         fibonacci(1, seq);
142     }
143
144
145
146 }
147
148 public class LargestProductSeries {
149
150     //Because this number is way too long, store it as a string.
151     private static String series =
"73167176531330624919225119674426574742355349194934969835203127745063262395783180169848018 69";
152
153     public static void main(String[] args) {
154
155
156         int consecutiveNum = 13;
157         long largestProduct = 0;
158
159             //for loop to iterate through the 13 consecutive terms in the large number
160           for (int i = 0; i < series.length() - consecutiveNum + 1; i ++)
161           {
162             long currentProduct = 1;
163
164             //for loop to iterate through the individual numbers in the 13 consecutive terms
and
165             //store the product as an integer using the .parseInt() method
166             for (int j = i; j < i + consecutiveNum; j++)
167             {
168                 currentProduct *= Integer.parseInt(series.substring(j, j + 1));
169             }
```

```
170
171             //if statement to set the largest product
172             if (currentProduct > largestProduct)
173             {
174                     largestProduct = currentProduct;
175                }
176           }
177           System.out.println(largestProduct);
178         }
179     }
```