```java
// Press Shift twice to open the Search Everywhere dialog and type `show whitespaces`,
// then press Enter. You can now see whitespace characters in your code.
import java.util.ArrayList;
public class Main {
    public static void main(String[] args) {
        // Press Opt+Enter with your caret at the highlighted text to see how
        // IntelliJ IDEA suggests fixing it.
        System.out.println("Answers:");
        System.out.println(sum35());
        System.out.println(fibonacci());
        System.out.println(prime());
        System.out.println(palindrome());
        System.out.println(smallest());
        System.out.println(sum());
        System.out.println(firstPrime());
        System.out.println(thirteen());
        pythagorean();
        System.out.println(sumOfPrimes());

        // Press Ctrl+R or click the green arrow button in the gutter to run the code.

    }
    public static int sum35(){
        int sum = 0;
        for (int i = 0; i < 1000; i++) {
            if(i%3 == 0 || i%5 == 0) {
                sum += i;
            }
        }
        return sum;
    }
    public static long fibonacci(){
        long fib = 0;
        long holder = 0;
        long fib0 = 1;
        long sum = 0;
        while(fib <=4000000) {
            if (fib % 2 == 0) {
                sum += fib;
            }
            holder = fib;
            fib += fib0;
            fib0 = holder;
        }
        return sum;
    }
    public static long prime(){
        long n = 600851475143L;
        long i = 1;
        while(i < n){
            if(n%i == 0){
                long div = n/i;
                if(isPrime(div)){
                    return div;
                }
            }
            i ++;
        }
        return 1;
    }
```

```java
public static boolean isPrime(long n) {
    for(long i = 2; i < n; i ++){
        if(n%i==0 && n != 0){
            return false;
        }
    }
    return true;
}
public static int palindrome(){
    int largest = 0;
    for(int i = 999; i > 0 ; i --){
        for(int j = i; j > 0; j --){
            if(isPalindrome(j*i)){
                if(j*i > largest){
                    largest = j*i;
                }
                break;
            }
        }
    }
    return largest;
}
public static boolean isPalindrome(int n){
    int reverse = 0;
    int num = n;
    while(num != 0){
        int digit = num%10;
        reverse  = reverse*10 + digit;
        num /=10;
    }
    if(reverse == n){
        return true;
    }
    return false;
}

public static long smallest(){
    long ans = 1;
    for(long i = 1; i <= 20; i ++){
       if(ans%i != 0) {
           ans = lcm(i, ans);
       }

    }
    return ans;
}
public static long lcm(long i, long n){

    long n0 = Math.max(i, n);
    long i0 = Math.min(i, n);
    long lcm = n0;
    while(lcm%i0 != 0){
        lcm+= n0;
    }
    return lcm;
}

public static long sum(){
    long sum = 0;
    long sum1 = 0;
```

```
        for(long i = 0; i <= 100; i ++){
            sum += i*i;
            sum1 += i;
        }
        sum1 *= sum1;
        return sum1- sum;
    }
    public static long firstPrime(){
        int counter = 0;
        long i = 2;
        while(counter < 10001){
            if(isPrime(i)){
                counter ++;
            }
            i ++;
        }
        return i - 1;
    }

    public static long thirteen(){
        String digit =
"731671765313306249192251196744265747423553491949349698352031277450632623957831801698480186947885184385861560789112949495459501737958331952853208805511125406987471585238630507156932909632952274430435576689664895044524452316173185640309871112172238311362229893423380308135336276614282806444486645238749303589072962904915604407723907138105158593079608667017242712188399879790879227492190169972088809377665727333001053367881220235421809751254540594752243525849077116705560136048395864467063244157221553975369781797784617406495514929086256932197846862248283972241375657056057490261407972968652414535100474821663704844031998900088952434506585412275886668811642717147992444292823086346567481391912316282458617866458359124566529476545682848912883142607690042242190226710556263211111093705442175069416589604080719840385096245544436298123098787992724428490918884580156166097919133875499200524063689912560717606058861164671094050775410022569831552000559357297257163626956188267042825248360082325753042075296345 0";
        long product = 1;
        for(int i = 0; i < 13; i ++){
            product *= Long.parseLong(digit.substring(i, i + 1));
        }
        for(int i = 1; i <987; i ++){
            long product2 = 1;
            for(int j = 0; j < 13; j ++){
                product2*= Long.parseLong(digit.substring(j + i, j + i + 1));
            }
            if(product2 > product){
               product = product2;
            }

        }
        return product;
    }
    public static long sumOfPrimes(){
        long sum = 2;
        ArrayList<Long> primes = new ArrayList<Long>();
        primes.add(2L);
        for(long i = 3; i < 2000000; i += 2){
            if(isPrime2(i, primes)){
                primes.add(i);
                sum += i;
            }
        }
        return sum;
    }
    public static boolean isPrime2(long n, ArrayList<Long> primes) {
```

```
        for(Long i: primes){
            if(n%i==0 && n != 2){
                return false;
            }
        }
        return true;
    }

    public static void pythagorean(){
        int n = 1000;
        for(int i = 1;i < 1000; i ++){
            for(int j = 1; j < 1000; j ++){
                for(int k = 1; k < 1000; k ++){
                    if(i+j+k == 1000 && (i*i)+(j*j) == (k*k)){
                        System.out.println(j*k*i);
                        return;
                    }
                }
            }
        }

    }

}
```