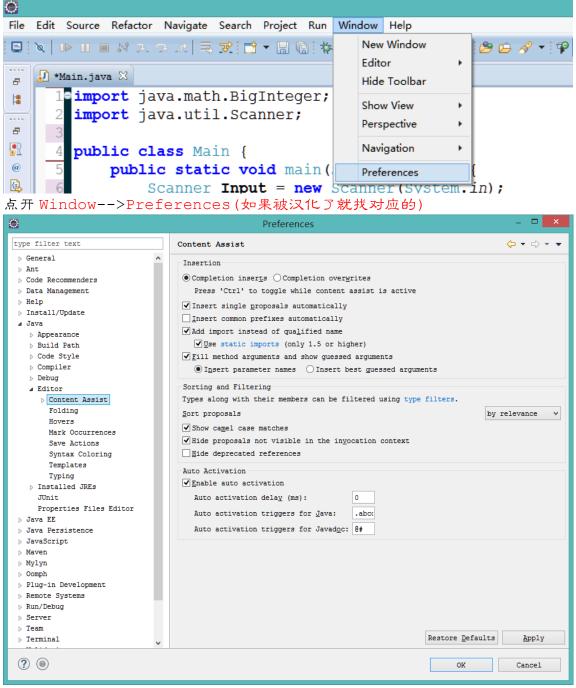
高精度

```
//2817 Palindrom Numbers
#include <iostream>
#include <string>
#include <stdlib.h>
#include <stdio.h>
using namespace std;
bool ispalindrom(int n, int base) {//这里把n变为base进制。如二
进制八进制等等, base从2到16
  int digit[100], cnt = 0;
  //这里应该不用写太详细了吧……进制转换, 存到数组digit中
  while (n != 0) {
     digit[cnt++] = n % base;
     n = n / base;
  for (int i = 0; i * 2 < cnt; i++) {//检查是不是回文
     if (digit[i] != digit[cnt - 1 - i]) {//如果不是
        return false;
     }
  return true;
int main() {
  int n;
  while (scanf("%d", &n) && n) {
//读入多组数据,用 while(scanf(...)), && n 代表 n!=0,如果为 0 则退
出
     int re[100], cnt = 0;//re数组存符合条件的base,cnt是个数
     for (int i = 2; i <= 16; i++) {//枚举所有的base 2~16
        if (ispalindrom(n, i))//如果当前base下n可以为回文数
           re[cnt++] = i; // 存到re数组里面
     if (cnt == 0) //如果cnt=0说明沒有符合条件的base
        printf("Number %d is not a palindrom\n", n);
     else {
        printf("Number %d is palindrom in basis", n);
        for (int i = 0; i < cnt; i++) {</pre>
          printf(" %d", re[i]);
        }
```

```
printf("\n");
     }
  return 0;
}
//2818 n!
//对于求一个很大很大的数的值的类型题,建议用java,C++会混麻烦
import java.math.BigInteger;//这是大整数类需要导入的包。类似于头文件
import java.util.Scanner;//这是输入需要导入的包
public class Main {//注意这里主类一定是Main,缺省包
  public static void main (String args[]) {//主方法名是main
     Scanner Input = new Scanner(System.in);//这句话是固定
的。只要你想输入, 当然Input是变量名可以改
     int t = Input.nextInt();//这代表读入一个整数
     BigInteger re = BigInteger.ONE; //声明一个BigInteger类的
对象, 并赋初始值为1, BigInteger.ONE 代表一个大整数1
     for(int i = 1;i <= t;i ++) {</pre>
        re = re.multiply(BigInteger.valueOf((long)i));
     System.out.println(re);
  }
}
//BigInteger.valueOf((long)i)是把整型 i 强制转化为长整型。然后转化
为大整数, BigInteger.valueOf(参数)是返回参数转化为大整数后的对象。参
数一定是长整型
注意:大整数类之间的运算不能通过+-*/%等等来进行,要通过这个大整数类自带
的函数来进行运算,具体如下(声明两个大整数类对象 b和 c):
BigInteger b,c;
         b.add(c)
                                     b + c
       b.multiply(c)
                                     b * c
        b.divide(c)
                                     b / c
       b.subtract(b)
                                     b - c
         b.mod(c)
                                     b % c
```

这些只是一些基本操作,单词记着混麻烦,到时候考试用 Eclipse 有自动补全,但是需要大家设置一下,因为初始设置是只有输入小数点才会提示,设置方式如下:



然后点击 Java——>Editor——>Content Assist,然后看右边出来的这一部

分

uto Activation Enable auto activation	
Auto activation delay (ms):	0
Auto activation triggers for <u>J</u> ava:	.abcc
Auto activation triggers for Javadoc:	@#

在红色框中输入大写字母从 A-Z, 小写字母 a-z, 一共 52 个字母,(不义按顺序,键盘上面一行一行的按就可以了), 设置好后点击 OK, 之后写代码会很方便 然后在键盘输入时候会发现有提示:

```
public class Main {
      public static void main(String args[]) {
             Scanner Input = new Scanner(System.in);
             int t = Input.nextInt();
             BigInteger b,c;
             b.
             b. • signum(): int - BigInteger - 0.42%
             Bi
                  equals(Object x) : boolean - BigInteger - 0.37%
             fo hashCode(): int - BigInteger - 0.27%
                   • abs(): BigInteger - BigInteger

    and(BigInteger val) : BigInteger - BigInteger

             Sy andNot(BigInteger val) : BigInteger - BigInteger
                   bitCount(): int - BigInteger
                   bitLength(): int - BigInteger
                   byteValue(): byte - Number
                   byteValueExact(): byte - BigInteger
                   o clearBit(int n): BigInteger - BigInteger
                   • divide(BigInteger val) : BigInteger - BigInteger

    divideAndRemainder(BigInteger val) : BigInteger[] - BigInteger

                   doubleValue(): double - BigInteger
                   • flipBit(int n) : BigInteger - BigInteger
                   • floatValue(): float - BigInteger
                   gcd(BigInteger val) : BigInteger - BigInteger
                   • getClass() : Class <?> - Object
                   getLowestSetBit(): int - BigInteger
                   o intValueExact(): int - BigInteger
                   isProbablePrime(int certainty): boolean - BigInteger
                   longValue(): long - BigInteger
                   longValueExact(): long - BigInteger
                   max(BigInteger val) : BigInteger - BigInteger
                   o min(BigInteger val) : BigInteger - BigInteger
                   mod(BigInteger m): BigInteger - BigInteger
                   • modInverse(BigInteger m) : BigInteger - BigInteger
                   o modPow(BigInteger exponent, BigInteger m): BigInteger - BigInteger
                   multiply(BigInteger val): BigInteger - BigInteger
                   negate(): BigInteger - BigInteger
                                                                                                    Press 'Alt+/' to show Template Proposals
```

这个时候你就下翻找你要的方法就可以了, 选好了按 Enter 键。具体的自己试一下

//2828 开方数

```
//这个题一点意思都沒有,背下来吧……
#include <cmath>
#include <cstdio>
```

```
#include <cstdlib>
#include <iostream>
#include <algorithm>
using namespace std;
int main() {
   double n,p;
   while (~scanf ("%lf%lf", &n, &p)) {//注意!!! 这里是读到文件结尾。
用~scanf("%lf%lf",&n,&p) 或者 ~scanf("%lf%lf",&n,&p)! = EOF
       if(n == 0 && p == 0)
                              break;
      printf("%.0f\n",pow(p,1 / n));//p开n次方就是p的1/n次方,
保留整数
   }
   return 0;
}
//2830 最小公倍数
//这个题依旧用 java 来写,因为最后 n == 100 时候,答案很大会溢出
import java.math.BigDecimal;
import java.math.BigInteger;
import java.util.Scanner;
public class Main {
   public static BigInteger gcd(BigInteger a, BigInteger b) {
   //这是求最大公约数的模板。照着写就行
     if(b.compareTo(BigInteger.ZERO) == 0) return a;//如果b == 0返回a
            return gcd(b, a.mod(b));//否则递归返回gcd(b, a.mod(b))
  public static void main(String args[]) {
     Scanner Cin = new Scanner(System.in);
     while(Cin.hasNext()) {//有多组数据, java用Cin.hasNext()代表读到文件结尾
        int n = Cin.nextInt();//读入整数n
        BigInteger re = BigInteger.ONE; //声明大整数re, 初始化为1
        for(int i = 2;i <= n;i ++) {</pre>
           BigInteger temp = BigInteger.valueOf(i);
            //把i转化为大整数赋值给temp
           BigInteger GCD = gcd(re,temp);//求re和temp的最大公约数
           re = re.multiply(temp).divide(GCD);
            //两个数(a,b)的最小公倍数等于a乘以b除以GCD
        System.out.println(re);
     }
  }
}
```