PLUS COMTONT

float和double

内容比之前题少就写在一个文件里了 float的范围为-2^128 ~ +2^128（-3.40E+38 ~ +3.40E+38） 存储格式为符号位**1bit**，指数阶码为**8bit**，尾数为**23bit** double的范围为-2^1024 ~ +2^1024（-1.79E+308 ~ +1.79E+308） 存储格式为符号位**1bit**，指数阶码为**11bit**，尾数为**52bit** 科学计数法的计算公式为$(-1)^S*F*2^E$ 其中S为符号位，F为尾数域，E为指数域 为了尽可能地充分利用各个bit，根据科学计数法的格式，二进制的科学计数法小数点前是1时才是标准的写法，例如$0.101 * 2^2$、 $10.101 * 2^2$,$0.101*2^2$就不是标准的计数方式，$1.01 * 2^1$、 $1.0101 * 2^3$就是标准的计数格式，所以省略小数点前的1，这样就可以用尽可能多的bit表示尾数域，从而提高浮点数的精度。 以float为例展示计算过程 **S=0，F全部是0，指数域全部是1**时为$+1.0_2*2^{255-127}$应的十进制约为：$+ 1.0 * 2 ^{+ 128} ≈ + 3.4 * 10 ^{38} $, **负的无穷大就是当S=1**的时候，即：$− 1.0 *2^{+ 128}≈ − 3.4 * 1 0^{38}$ 值为$+1.0*2 ^{−127}$ double的计算与此同理

参考https://blog.csdn.net/qq_42144047/article/details/120584932

String

有两种创建方式

```
//字面量的方式定义
String name1 = "man!";
//创建对象的方式定义
String name2 = new String("what can i say");
```

在JDK8版本中，使用的是char字符数组的方式实现这个String类型的字符串，在JDK9版本开始，实现这个String字符串的方式是使用byte数组实现,原因是char是两个字节，String只占一个，所以会浪费很大空间 **String代表不可变的字符序列，简称不变性**

代码输出结果

```
method: hello world
main: hello
```

原因：一旦字符串被创建，它的内容就不能被改变，执行s += " world";时，实际上创建的是一个新对象，原有字符串内容没有改变，所以打印main函数里的s只会输出hello。

Task1.变量和数据类型

1

1. 整型：byte short int long

2. 字符型：char
3. 浮点型：float double
4. 布尔型：boolean（记得c是bool吧）

2

1. **byte** -128~127 8位带符号数
2. **short** -32768~32767 16位带符号数
3. **int** -2的31次方~2的31次方-1 32位带符号数
4. **long** -2的63次方~2的63次方-1 64位带符号数

3

```
int a=4
char c='0';
int b=a+c;
```

发生了**自动类型转换**，b=52，因为'0'为ascii编码，整数为48，和a相加时就转化为了int整型即4+48=52.（和c语言一样）（自己也发现一般java范围小类型向范围大类型转换就会自动进行，即**自动类型转换**，而大范围向小范围转换如double变int就必须强制使用类型转换函数，即**强制类型转换**，应用时很大概率会丢失精度，或者我认为丢失了一部分原有的信息）（区别就是自动很方便，不用一直写转换函数，但是容易被坑，有时候死活找不到问题）（强制比较灵活一点，很好的满足自己的需要，就是打字起来比较烦）

4

1. **包装类**,就是将基本数据类型和一些辅助方法包装到类中,因为Java有对象这个概念，基本数据类型可能不能很好完成与对象相关的操作，于是就产生了包装类，题目中的Integer就是包装类，
2. **引用类型**，是用于存储对象引用的数据类型，通过引用类型的变量，我们可以间接操作对象，声明变量时，存储的是对象在内存的一个地址（指针？）。
3. **基本数据类型缓存池**以integer为例，范围为-128~127，使用Integer.valueOf(18)因为18在范围内，于是直接从缓存池池调用了一个对象，若我再使用Integer.valueOf(18)，继续调用一个对象，由于18值一样，两次调用的对象是同一个，也就是地址一样

代码解释

```
Integer x = new Integer(18);
Integer y = new Integer(18);
System.out.println(x == y);
```

创造两个对象x，y（**他们不一样**） x==y来验证他们地址一不一样，结果可想而知为false

```
Integer z = Integer.valueOf(18);
Integer k = Integer.valueOf(18);
System.out.println(z == k);
```

从缓存池里调用，声明为两个z，k 并进行验证如上文所述，地址一样，结果为true

```
Integer m = Integer.valueOf(300);
Integer p = Integer.valueOf(300);
System.out.println(m == p);
```

300不在缓存池范围内，此时变为创建一个新对象，**值得注意的是**，创建的m，p不是一个对象，地址并不一样，所以输出结果为false

Task2.运算符

结果为

```
13
6 8
```

代码解释

1. 定义a=5，b=7，类型都为int
2. 定义c= (++a) + (b++)，注意到++a意思为执行命令前a就已经加1，b++为执行完命令后b再加1；所以c的值就等于6(a)+7(b)=13，并且b+1后b的值为8
3. 输出结果c=13，a=6，b=8，a和b中间还有个空格

补码

a&(-a) 的二进制形式是0010， 原理：-a是a的补码，即a取反后再加1，所以我们先取反，即a的0变1，1变0，即**1101**.+1后变为**1110**即为-a，&运算指的是两个为1结果为1，否则为0；所以a&(-a)=**0010**，我就发现a&(-a)求的是二进制中最小1的位置。

总结

由于事先学过一点c++,所以运算符什么的很快就接受了，做这题看的最久的还是java特有的对象等等内容（特别是拓展的4，一个名词一个名词的查去了解内容，意思，作用），学到了很多知识，最后我也去了解了树状数组的内容