1

补全函数

```
//这个函数用于判断传入的年份是否为闰年
//是闰年返回1，不是闰年返回2
boolean isLeapYear(int year){
    if((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)){
        return 1;
    }else{
        return 2;
    }
}
```

switch-case语句判断一个变量与一系列值中某个值是否相等，每个值称为一个分支，用于根据变量的值执行不同的代码块。

不正确 虽然用switch-case在功能上与if-else相似，编写更简单更可读，但底层实现原理不同。 switch-case会生成一个跳转表来指示实际的case分支的地址，if-else是遍历条件分支直到命中条件。 在使用时，需要对同一个变量进行大量相等性判断时，优先使用 switch-case，代码更清晰且性能更好；当条件判断涉及范围比较、多个变量或复杂逻辑时，使用if-else。

2

代码补全

```
import java.util.Scanner;
public class text {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        text t=new text();
        t.print(n);
    }
    void print(int n){
        int m=n/2;
        int i=0;
        while(i<n){
            int d=Math.abs(i-m);
            int j=0;
            while(j<d){
                System.out.print(" ");
                j++;
            }
            if(i==0||i==n-1){
                System.out.print("*");
```

```
            }else{
                System.out.print("*");
                int s=n-2*d-2;
                int k=0;
                while(k<s){
                    System.out.print(" ");
                    k++;
                }
                System.out.print("*");
            }
            System.out.println();
            i++;
        }
    }
}
```

运行

```
PS D:\IT\VS\Cuse> javac text.java
PS D:\IT\VS\Cuse> java text.java
5
  *
 * *
*   *
 * *
  *
PS D:\IT\VS\Cuse> java text.java
7
   *
  * *
 *   *
*     *
 *   *
  * *
   *
```

3

迭代

```
import java.util.Scanner;
public class text {
    public static int fibonacci(int n){
        if(n<=1){
            return n;
        }
        int P1=0;
        int P2=1;
```

```
        int R=0;
        for(int i=2;i<=n;i++){
            R=P1+P2;
            P1=P2;
            P2=R;
        }
        return R;
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        System.out.println(fibonacci(n));
    }
}
```

```
PS D:\IT\VS\Cuse> javac text.java
PS D:\IT\VS\Cuse> java text.java
10
55
PS D:\IT\VS\Cuse> java text.java
5
5
PS D:\IT\VS\Cuse>
```

递归

```
import java.util.Scanner;
public class text {
    public static int fibonacci(int n){
        if(n<=1){
            return n;
        }
        return fibonacci(n-1)+fibonacci(n-2);
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        System.out.println(fibonacci(n));
    }
}
```

```
PS D:\IT\VS\Cuse> javac text.java
PS D:\IT\VS\Cuse> java text.java
1
1
PS D:\IT\VS\Cuse> java text.java
7
13
PS D:\IT\VS\Cuse>
```

迭代的核心思想是重复步骤，通过循环的机制解决问题；递归的核心思想是将问题拆分处理，通过函数自调用与返回栈解决问题。

偏好迭代是因为：迭代没有调用函数，性能会更好；迭代常使用固定且少量的内存，不会像递归在多次调用后耗尽栈内存。

不能完全取代 虽然能实现一样的功能，但与循环相比，递归的效率更低，风险更高。

4

```java
import java.util.Scanner;
//根据规则，大的盘子只能在小的盘子下，则对于一个在A柱的第n个盘子，必须先将第n-1个盘子移到B柱才能将第n个盘子移到C柱。确定递归方式。
public class text {
    void hanoi(int n){
        M(n,'A', 'B', 'C');//递归函数调用
    }
    void M(int n,char A,char B,char C){
        if(n==1){
            System.out.println(A+"->"+C);//只有一个盘子时，直接从A移动到C
        }else{
            M(n-1,A,C,B);//将上面的n-1个盘子从A借助C移动到B
            System.out.println(A+"->"+C);//再将第n个盘子从A移动到C
            M(n-1,B,A,C);//最后将B上的n-1个盘子借助A移动到C
        }
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        text t=new text();
        t.hanoi(n);
        //用于输入n的值
    }
}
```

```
PS D:\IT\VS\Cuse> javac text.java
PS D:\IT\VS\Cuse> java text.java
3
A->C
A->B
C->B
A->C
B->A
B->C
A->C
```

```
PS D:\IT\VS\Cuse> java text.java
5
A->C
A->B
C->B
A->C
B->A
B->C
A->C
A->B
C->B
C->A
B->A
C->B
A->C
A->B
C->B
A->C
B->A
B->C
A->C
B->A
C->B
C->A
B->A
B->C
A->C
A->B
C->B
A->C
B->A
B->C
A->C
PS D:\IT\VS\Cuse>
```