湖南省高等职业院校学生专业技能考核题库

(软件技术专业)

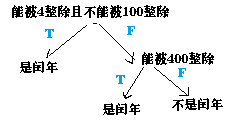
**模块二 应用程序测试**

**1.试题编号：H2-1《人力资源管理系统》应用程序单元测试**

任务一：判断闰年

**解答：**

①根据本模块附件录1设计测试用例表的要求分别设计测试用例。(20分)

表 1 设计测试用例

|  |  |
| --- | --- |
| 用例要求 | 测试用例 |
| 语句覆盖 | year=2004  year=2000  year=2001 |
| 判定覆盖 | year=2004  year=2000  year=2100 |
| 条件覆盖 | year=2000  year=2100  year=2001 |
| 判定/条件覆盖 | year=2004  year=2000  year=2100  year=2001 |
| 组合覆盖 | year=2004  year=2100  year=2000  year=2001 |
| 路径覆盖 | year=2004  year=2000  year=2100 |

②使用 xunit 工具和路径覆盖测试用例编写单元测试程序，展示测试结果，分析测试结果。(10分)

类代码：

**public** **class** Year {

**public** **boolean** isLeepYear(**int** year){

//能被4整除且不能被100整除

**if**((year%4==0)||(year%100!=0)){

**return** **true**;

//能被400整除

}**else** **if**(year%400==0){

**return** **true**;

}**else**{

**return** **false**;

}

}

测试类代码：

import static org.junit.Assert.\*;

import org.junit.Test;

public class YearTest {

@Test

public void test() {

Year y=new Year();

assertEquals(y.isLeepYear(2000),true);

assertEquals(y.isLeepYear(2100),false);

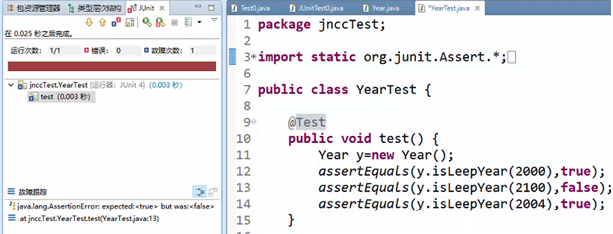
assertEquals(y.isLeepYear(2004),true);

assertEquals(y.isLeepYear(2001),false);

}

}

测试结果：有BUG



用例测试结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例要求 | 测试用例 | 预测结果 | 运行结果 | 结论 |
| 路径覆盖 | year=2004 | True | True |  |
| year=2000 | True | True |  |
| year=2100 | false | True | BUG |

分析BUG原因：2100年能被4整除，也能被100整除，正确判断为不是闰年

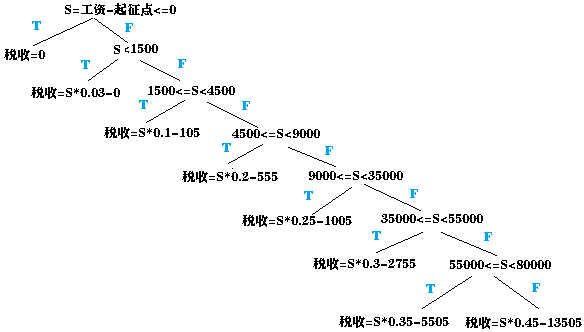
测试2100年属于闰年，从程序year%100!=0前后找原因，原来是||或运算符的原因，正确符号应该是&&与运算符才对。

**if**((year%4==0)||(year%100!=0))错误

正确的应该是**if**((year%4==0)&&(year%100!=0))

任务二：计算工资纳税额

1. 根据本模块附件录1设计测试用例表的要求分别设计测试用例。(20分)

表 1 设计测试用例

|  |  |
| --- | --- |
| 用例要求 | 测试用例 |
| 语句覆盖 | Salary=3500,deduct=3500  Salary=4999,deduct=3500  Salary=5000,deduct=3500  Salary=8000,deduct=3500  Salary=12500,deduct=3500  Salary=38500,deduct=3500  Salary=58500,deduct=3500  Salary=83500,deduct=3500 |
| 判定覆盖 | Salary=3500,deduct=3500  Salary=4999,deduct=3500  Salary=5000,deduct=3500  Salary=8000,deduct=3500  Salary=12500,deduct=3500  Salary=38500,deduct=3500  Salary=58500,deduct=3500  Salary=83500,deduct=3500 |
| 条件覆盖 | Salary=3500,deduct=3500  Salary=4999,deduct=3500  Salary=5000,deduct=3500  Salary=8000,deduct=3500  Salary=12500,deduct=3500  Salary=38500,deduct=3500  Salary=58500,deduct=3500  Salary=83500,deduct=3500 |
| 判定/条件覆盖 | Salary=3500,deduct=3500  Salary=4999,deduct=3500  Salary=5000,deduct=3500  Salary=8000,deduct=3500  Salary=12500,deduct=3500  Salary=38500,deduct=3500  Salary=58500,deduct=3500  Salary=83500,deduct=3500 |
| 组合覆盖 | Salary=3500,deduct=3500  Salary=4999,deduct=3500  Salary=5000,deduct=3500  Salary=8000,deduct=3500  Salary=12500,deduct=3500  Salary=38500,deduct=3500  Salary=58500,deduct=3500  Salary=83500,deduct=3500 |
| 路径覆盖 | Salary=3500,deduct=3500  Salary=4999,deduct=3500  Salary=5000,deduct=3500  Salary=8000,deduct=3500  Salary=12500,deduct=3500  Salary=38500,deduct=3500  Salary=58500,deduct=3500  Salary=83500,deduct=3500 |

public class Salary {

/\*\*

\*个人所得税具体计算方法

\*注意：本程序中假设工资已经减去五险一金

\*@param salary工资

\*@param deduct起征点

\*@return 个人所得税

\*/

public double selfValue(int salary,int deduct){

double sefValue=0;

if(salary<1500){

sefValue=(double)(salary-deduct)\*0.03-0;

}else if(salary>=1500&&salary<4500){

sefValue=(double)(salary-deduct)\*0.1-105;

}else if(salary>=4500&&salary<9000){

sefValue=(double)(salary-deduct)\*0.2-555;

}else if(salary>=9000&&salary<35000){

sefValue=(double)(salary-deduct)\*0.25-1005;

}else if(salary>=35000&&salary<55000){

sefValue=(double)(salary-deduct)\*0.3-2755;

}else if(salary>=55000&&salary<80000){

sefValue=(double)(salary-deduct)\*0.35-5505;

}else {

sefValue=(double)(salary-deduct)\*0.45-13505;

}

return sefValue;

}

}

②使用 xunit 工具和路径覆盖测试用例编写单元测试程序，展示测试结果，分析测试结果。(10分)

import static org.junit.Assert.\*;

import org.junit.Test;

public class TestSalary {

@Test

public void test() {

Salary S=new Salary();

assertEquals((Double)S.selfValue(3500, 3500),(Double)0.0);

assertEquals((Double)S.selfValue(4999, 3500),(Double)44.97);

assertEquals((Double)S.selfValue(5000, 3500),(Double)45.0);

assertEquals((Double)S.selfValue(8000, 3500),(Double)345.0);

assertEquals((Double)S.selfValue(12500, 3500),(Double)1245.0);

assertEquals((Double)S.selfValue(38500, 3500),(Double)7745.0);

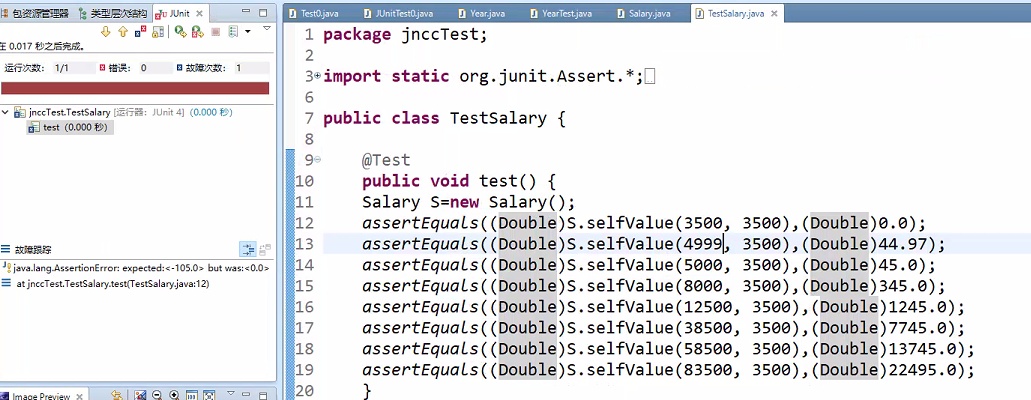
assertEquals((Double)S.selfValue(58500, 3500),(Double)13745.0);

assertEquals((Double)S.selfValue(83500, 3500),(Double)22495.0);

}

}

测试结果：有BUG



用例测试结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例要求 | 测试用例 | 预测结果 | 运行结果 | 结论 |
| 路径覆盖 | salary=3500 | 0 | 105 | Bug |
| salary=4999 | 44.97 | 255.2 | Bug |
| salary=5000 | 45 | 255 | Bug |
| salary=8000 | 345 | 345 |  |
| salary=12500 | 1245 | 1245 |  |
| salary=38500 | 7745 | 7745 |  |
| salary=58500 | 13745 | 13745 |  |
| salary=83500 | 22495 | 22495 |  |

分析BUG原因：salary小于3500的起征点，应该不用收税，程序不应该以salary判断，应该以salary-deduct判断，且应该增加没有salary-deduct<=0的情况。

有BUG的代码：

if(salary<1500){

sefValue=(double)(salary-deduct)\*0.03-0;

正确的代码：

if(salary-deduct<=0){

sefValue=0;

}Else if(salary-deduct<1500){

sefValue=(double)(salary-deduct)\*0.03-0;

任务三：信息服务功能  
 验证 18 位身份证号码并判断身份证主人的性别，身份证号码的规则为：

前17位全部由数字组成，最后一位为数字或者字符'x' —个字符cH为数字的  
条件为：cH>=' 0' && cH<=' 9'；

**import** java.util.Scanner;

**public** **static** **void** func(){

**char**[] card=**new** **char**[18];

Scanner input=**new** Scanner(System.*in*);

System.*out*.println("请输入一个18位身份证号码：");

String s=input.next();

input.close();

**if**(s.length()!=18){

System.*err*.println("输入错误：要求输入的身份证号码为18位");

**return**;

}

**for**(**int** i=0;i<card.length;i++){

card[i]=s.charAt(i);

**if**(i==18){//判断最后一位

**if**((card[i]>'9'&&card[i]<'0')||card[i]!='x'){

System.*err*.println("身份证号码输入有误");

**return**;

}

}**else** **if**(card[i]>'9'||card[i]<'0'){

System.*err*.println("身份证号码输入有误");

**return**;

}

}

**int** an=Integer.*parseInt*(String.*valueOf*(card[16]));

System.*out*.println(an);

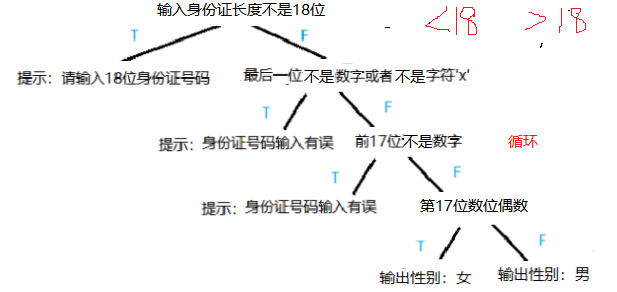
**if**(card[16]%2==0)

System.*out*.println("女");

**else**

System.*out*.println("男");

}

①前17位数为奇数表示性别为男，偶数表示性别为女。  
 从键盘输入一个 18位的身份证号码保存到字符数组 card 中。  
 输出：主人性别。

①根据本模块附件录 1 设计测试用例表的要求分别设计测试用例。（20 分）

表 1 设计测试用例

|  |  |
| --- | --- |
| 用例要求 | 测试用例 |
| 语句覆盖 | 12345678901234567  12345678901234567a  1234567890123456x8  123456789012345678  123456789012345668 |
| 判定覆盖 | 12345678901234567  12345678901234567a  1234567890123456x8  123456789012345678  123456789012345668 |
| 条件覆盖 | 12345678901234567  1234567890123456789  12345678901234567a  12345678901234567X  1234567890123456x8  123456789012345678  12345678901234568x |
| 判定/条件覆盖 | 12345678901234567  1234567890123456789  12345678901234567a  12345678901234567X  1234567890123456x8  123456789012345678  12345678901234568x |
| 组合覆盖 | 12345678901234567  1234567890123456789  12345678901234567a  12345678901234567X  12345678901234567x  123456789012345678  1234567890123456x8  123456789012345688 |
| 路径覆盖 | 12345678901234567  1234567890123456789  12345678901234567a  //下面是循环语句的路径覆盖  x23456789012345678  1x34567890123456x8  12x456789012345678  123x56789012345678  1234x6789012345678  12345x789012345678  123456x89012345678  1234567x9012345678  12345678x012345678  123456789x12345678  1234567890x2345678  12345678901x345678  123456789012x45678  1234567890123x5678  12345678901234x678  123456789012345x78  1234567890123456x8  x2345678901234567x  1x345678901234567x  12x45678901234567x  123x5678901234567x  1234x678901234567x  12345x78901234567x  123456x8901234567x  1234567x901234567x  12345678x01234567x  123456789x1234567x  1234567890x234567x  12345678901x34567x  123456789012x4567x  1234567890123x567x  12345678901234x67x  123456789012345x7x  1234567890123456xx  //循环体结束  123456789012345668  12345678901234566x  123456789012345678  12345678901234567x |

②使用 xUnit 工具和路径覆盖测试用例编写单元测试程序，展示测试结果，分析测试结果。（10 分）

Junit单元测试代码：

**import** **static** org.junit.Assert.\*;

**import** org.junit.Test;

**public** **class** TestShenFenZheng {

@Test

**public** **void** testFunc() {

ShenFenZheng.*func*();

}

}

用例测试结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例要求 | 测试用例 | 预测结果 | 运行结果 | 结论 |
| 路径覆盖 | 12345678901234567 | 请输入18位身份证号码 | 请输入18位身份证号码 |  |
| 1234567890123456789 | 请输入18位身份证号码 | 请输入18位身份证号码 |  |
| 12345678901234567a | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 123456789012345668 | 6  女 | 6  女 |  |
| 12345678901234567x | 7  男 | 身份证号码输入有误 | BUG |
| x23456789012345678 | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 1x34567890123456x8 | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 12x456789012345678 | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 123x56789012345678 | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 1234x6789012345678 | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 12345x789012345678 | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 123456x89012345678 | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 1234567x9012345678 | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 12345678x012345678 | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 123456789x12345678 | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 1234567890x2345678 | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 12345678901x345678 | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 123456789012x45678 | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 1234567890123x5678 | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 12345678901234x678 | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 123456789012345x78 | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 1234567890123456x8 | 身份证号码输入有误 | 身份证号码输入有误 |  |
|  | x2345678901234567x | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 1x34567890123456xx | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 12x45678901234567x | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 123x5678901234567x | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 1234x678901234567x | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 12345x78901234567x | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 123456x8901234567x | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 1234567x901234567x | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 12345678x01234567x | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 123456789x1234567x | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 1234567890x234567x | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 12345678901x34567x | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 123456789012x4567x | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 1234567890123x567x | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 12345678901234x67x | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 123456789012345x7x | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 1234567890123456xx | 身份证号码输入有误 | 身份证号码输入有误 |  |
| 12345678901234566x | 6  女 | 身份证号码输入有误 | BUG |
| 123456789012345678 | 7  男 | 7  男 |  |

分析测试BUG原因：

12345678901234567、123456789012345668可以正常输出，12345678901234567x、12345678901234566x提示“身份证号码输入有误”，说明输入的最后一位为字符x时判断有问题。

有BUG的代码：

if(i==18){//判断最后一位

**if**((card[i]>'9'&&card[i]<'0')||card[i]!='x'){

System.*err*.println("身份证号码输入有误");

正确代码：

if(i==17){//判断最后一位

**if**((card[i]>'9'&&card[i]<'0')){

System.*err*.println("身份证号码输入有误");

**return**;

}**else** **if**(card[i]!='x'){

System.*err*.println("身份证号码输入有误");

**return**;

}

**2.试题编号：H2-2 《“帮你算”系统》应用程序单元测试**

（1）任务描述  
 随着网络的不断发展，我们每天接触的新鲜事物都在不断增加，处在这一个信息量大爆炸的时代,生活中有大量的计算问题，某公司准备开发一套“帮你算”系统，用来解决生活中简单、繁琐的数学问题。

现已经完成了相应功能代码，请按任务要求完成单元测试，如果程序有错误，报告错误，并提出修改意见。

任务一：计算平均成绩功能

已知某个班有n（最多30）个学生，学习m（最多5）门课程，已知所有学生的各科成绩。  
请编写程序：分别计算每个学生的平均成绩，并输出。

**import** java.util.Scanner;

**public** **static** **void** Student(){

Scanner sc=**new** Scanner(System.*in*);

**double**[][] N=**new** **double**[30][5];

**double**[] M=**new** **double**[30];

**double** sum=0;

**for**(**int** i=0;i<=3;i++){

sum=0;

**for**(**int** j=0;j<=5;j++){

N[i][j]=sc.nextFloat();

sum+=N[i][j];

}

M[i]=sum/5;

}

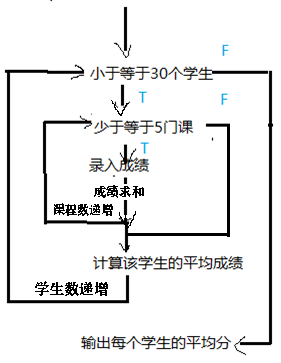
**for**(**int** i=0;i<=3;i++){

System.*out*.println("第"+i+"个学生的平均分为："+M[i]);

}

}

①根据本模块附件录1设计测试用例表的要求分别设计测试用例。（20分）

表 1 设计测试用例

|  |  |
| --- | --- |
| 用例要求 | 测试用例 |
| 语句覆盖 | 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4  1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5 |
| 判定覆盖 | 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5  1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4  1 1 1 1 1 1  1 1 1 1 1 |
| 条件覆盖 | 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5  1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4  1 1 1 1 1  1 1 1 1 1 1 |
| 判定/条件覆盖 | 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5  1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4  1 1 1 1 1  1 1 1 1 1 1 |
| 组合覆盖 | 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5  1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4  1 1 1 1 1  1 1 1 1 1 1 |
| 路径覆盖 | 1  1 1  1 1 1  1 1 1 1  1 1 1 1 1  1 1 1 1 1 2  1 1 1 1 1 2 2  1 1 1 1 1 2 2 2  1 1 1 1 1 2 2 2 2  1 1 1 1 1 2 2 2 2 2  1 1 1 1 1 2 2 2 2 2 3  1 1 1 1 1 2 2 2 2 2 3 3  1 1 1 1 1 2 2 2 2 2 3 3 3  1 1 1 1 1 2 2 2 2 2 3 3 3 3  1 1 1 1 1 2 2 2 2 2 3 3 3 3 3  1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4  1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4  1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4  1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4  1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4  1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5  1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5  1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5  1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5  1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5 |

②使用 xunit 工具和路径覆盖测试用例编写单元测试程序，展示测试结果，分析测试结果（10分）

JUnit单元测试代码：

import static org.junit.Assert.\*;

import org.junit.Test;

public class StudentAvgTest {

@Test

public void testStudent() {

StudentAvg.Student();

}

}

用例测试结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例要求 | 测试用例 | 预测结果 | 运行结果 | 结论 |
| 路径覆盖 |  | 无运行结果，等待继续输入 | 无运行结果，等待继续输入 |  |
| 1 | 无运行结果，等待继续输入 | 无运行结果，等待继续输入 |  |
| 1 1 | 无运行结果，等待继续输入 | 无运行结果，等待继续输入 |  |
| 1 1 1 | 无运行结果，等待继续输入 | 无运行结果，等待继续输入 |  |
| 1 1 1 1 | 无运行结果，等待继续输入 | 无运行结果，等待继续输入 |  |
|  | 1 1 1 1 1 | 无运行结果，等待继续输入 | 无运行结果，等待继续输入 |  |
| 1 1 1 1 1 2 | 无运行结果，等待继续输入 | 报错  ArrayIndexOutOfBoundException:5 | BUG |
| 1 1 1 1 1 2 2 | 无运行结果，等待继续输入 | 报错  ArrayIndexOutOfBoundException:5 | BUG |
| 1 1 1 1 1 2 2 2 | 无运行结果，等待继续输入 | 报错  ArrayIndexOutOfBoundException:5 | BUG |
| 1 1 1 1 1 2 2 2 2 | 无运行结果，等待继续输入 | 报错  ArrayIndexOutOfBoundException:5 | BUG |
| 1 1 1 1 1 2 2 2 2 2 | 无运行结果，等待继续输入 | 报错  ArrayIndexOutOfBoundException:5 | BUG |
| 1 1 1 1 1 2 2 2 2 2 3 | 无运行结果，等待继续输入 | 报错  ArrayIndexOutOfBoundException:5 | BUG |
| 1 1 1 1 1 2 2 2 2 2 3 3 | 无运行结果，等待继续输入 | 报错  ArrayIndexOutOfBoundException:5 | BUG |
| 1 1 1 1 1 2 2 2 2 2 3 3 3 | 无运行结果，等待继续输入 | 报错  ArrayIndexOutOfBoundException:5 | BUG |
| 1 1 1 1 1 2 2 2 2 2 3 3 3 3 | 无运行结果，等待继续输入 | 报错  ArrayIndexOutOfBoundException:5 | BUG |
| 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 | 无运行结果，等待继续输入 | 报错  ArrayIndexOutOfBoundException:5 | BUG |
| 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 | 无运行结果，等待继续输入 | 报错  ArrayIndexOutOfBoundException:5 | BUG |
| 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 | 无运行结果，等待继续输入 | 报错  ArrayIndexOutOfBoundException:5 | BUG |
| 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 | 无运行结果，等待继续输入 | 报错  ArrayIndexOutOfBoundException:5 | BUG |
| 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 | 无运行结果，等待继续输入 | 报错  ArrayIndexOutOfBoundException:5 | BUG |
| 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 | 第0个学生的平均分位1  第1个学生的平均分位2  第2个学生的平均分位3  第3个学生的平均分位4 | 报错  ArrayIndexOutOfBoundException:5 | BUG |
| 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 | 第0个学生的平均分位1  第1个学生的平均分位2  第2个学生的平均分位3  第3个学生的平均分位4 | 报错  ArrayIndexOutOfBoundException:5 | BUG |
| 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 | 第0个学生的平均分位1  第1个学生的平均分位2  第2个学生的平均分位3  第3个学生的平均分位4 | 报错  ArrayIndexOutOfBoundException:5 | BUG |
| 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 | 第0个学生的平均分位1  第1个学生的平均分位2  第2个学生的平均分位3  第3个学生的平均分位4 | 报错  ArrayIndexOutOfBoundException:5 | BUG |
| 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 | 第0个学生的平均分位1  第1个学生的平均分位2  第2个学生的平均分位3  第3个学生的平均分位4 | 报错  ArrayIndexOutOfBoundException:5 | BUG |
| 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5 | 第0个学生的平均分位1  第1个学生的平均分位2  第2个学生的平均分位3  第3个学生的平均分位4 | 报错  ArrayIndexOutOfBoundException:5 | BUG |

分析BUG原因：查看错误提示：ArrayIndexOutOfBoundException:5说明是数组超出界限了,输入数字超过5个的都报这个错误，检出有BUG的代码：

**for**(**int** j=0;j<=5;j++){

N[i][j]=sc.nextFloat();

sum+=N[i][j];

循环了6次，与**double**[][] N=**new** **double**[30][5]语句定义的数组大小相冲突

正确代码：

**for**(**int** j=0;j<=4;j++){

N[i][j]=sc.nextFloat();

sum+=N[i][j];

任务二：阶乘计算功能

实现n!功能。其中n小于10。

**import** java.util.Scanner;

**public** **static** **void** func(){

**long** sum=1;

Scanner sc =**new** Scanner(System.*in*);

**int** n=sc.nextInt();

**if**(n>=10)

System.*out*.println("输入的值不能超过10！");

**else**{

**for**(**int** i=1;i<n;i++){

sum\*=i;

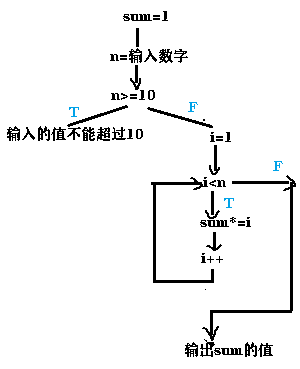
}

System.*out*.println(sum);

}

}

①根据本模块附件录1设计测试用例表的要求分别设计测试用例。(20分)

表 1 设计测试用例

|  |  |
| --- | --- |
| 用例要求 | 测试用例 |
| 语句覆盖 | n=10  n=2 |
| 判定覆盖 | n=10  n=2 |
| 条件覆盖 | n=10  n=2 |
| 判定/条件覆盖 | n=10  n=2 |
| 组合覆盖 | n=10  n=2 |
| 路径覆盖 | n=10  n=0  n=1  n=2  n=3  n=4  n=5  n=6  n=7  n=8  n=9 |

②使用 xunit 工具和路径覆盖测试用例编写单元测试程序，展示测试结果，分析测试结果。(10分)

JUnit单元测试代码：

**import** org.junit.Test;

**public** **class** JieChengTest {

@Test

**public** **void** testFunc() {

JieCheng.*func*();

}

}

用例测试结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例要求 | 测试用例 | 预测结果 | 运行结果 | 结论 |
| 路径覆盖 | n=10 | 输入的值不能超过10！ | 输入的值不能超过10！ |  |
| n=1 | 1 | 1 | bug |
| n=2 | 2 | 1 | bug |
| n=3 | 6 | 2 | bug |
| n=4 | 24 | 6 | bug |
| n=5 | 120 | 24 | bug |
| n=6 | 720 | 120 | bug |
| n=7 | 5040 | 720 | bug |
| n=8 | 40320 | 5040 | bug |
| n=9 | 362880 | 40320 | bug |
|  | N=0 | 1 | 1 |  |

分析BUG原因：

n!阶乘的实际结果正好等于（n-1）!阶乘的结果，推断应该是循环结束条件出现BUG。

下面代码存在BUG：

**for**(**int** i=1;i<n;i++){

sum\*=i;

}

正确的代码应该是：

**for**(**int** i=1;i<=n;i++){

sum\*=i;

}

任务三：前项列和计算功能

有一分数序列：2/1, 3/2, 5/3, 8/5, 13/8, 21/13… 求出这个数列的前n (n小于10)项之和。(34/21,55/34,89/55,144/89,233/144)

**import** java.util.Scanner;

**public** **static** **void** func(){

Scanner sc=**new** Scanner(System.*in*);

**double** fi,fu,swap,sum=0;

**int** n=sc.nextInt();

fi=2.0;

fu=1.0;

**for**(**int** i=1;i<=n;i++){

swap=fi;

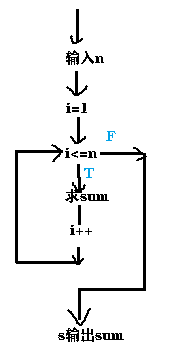
fi+=fu;

fu=swap;

sum+=fi/fu;

}

System.*out*.println(sum);

}

①根据本模块附件录1设计测试用例表的要求分别设计测试用例。(20分)

表 1 设计测试用例

|  |  |
| --- | --- |
| 用例要求 | 测试用例 |
| 语句覆盖 | n=1 |
| 判定覆盖 | n=2 |
| 条件覆盖 | n=3 |
| 判定/条件覆盖 | n=4 |
| 组合覆盖 | n=5 |
| 路径覆盖 | n=0  n=1  n=2  n=3  n=4  n=5  n=6  n=7  n=8  n=9  n=10  n=11 |

②使用 xunit 工具和路径覆盖测试用例编写单元测试程序,展示测试结果,分析测试结果。(10分)

JUnit单元测试代码：

**public** **class** FenShuXuLieTest {

@Test

**public** **void** testFunc() {

FenShuXuLie.*func*();

}

}

用例测试结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例要求 | 测试用例 | 预测结果 | 运行结果 | 结论 |
| 路径覆盖 | n=0 | 0 | 0 |  |
| n=1 | 2.0 | 2.0 |  |
| n=2 | 3.5 | 3.5 |  |
| n=3 | 5.166666666666667 | 5.166666666666667 |  |
| n=4 | 6.766666666666667 | 6.766666666666667 |  |
| n=5 | 8.391666666666667 | 8.391666666666667 |  |
| n=6 | 10.007051282051282 | 10.007051282051282 |  |
| n=7 | 11.626098901098901 | 11.626098901098901 |  |
| n=8 | 13.24374595992243 | 13.24374595992243 |  |
| n=9 | 14.86192777810424869 | 14.86192777810424869 |  |
| n=10 | 16.47990530619413633 | 16.47990530619413633 |  |
| n=11 | 18.09796086174969188 | 18.09796086174969188 |  |

分析BUG原因：

程序运行结果与预测的一致，但是没有限制n的取值，根据题意，n<10,对于n>=10需要给出提示。

下面代码存在BUG：

**int** n=sc.nextInt();

正确的代码应该是：

**int** n=sc.nextInt();

**if**(n>=10){

System.*out*.println("请输入小于10的数字");

}**else**{

（2）作品提交要求见本模块附录 2

（3）实施条件要求见本模块附录 3

（4）评价标准见本模块附录 4

**3.试题编号：H2-3《图形体积计算系统》应用程序单元测试**

(1)任务描述  
 图形在我们的生活中无处不在,在我们的周围,到处都是图形的缩影、例如空调是长方形、水瓶瓶盖是圆形,这些图形的计算对于土木工程师来说,非常重要。某公司开发出一套图形面积计算系统,来计算图形面积。现已经完成了相应功能代码,请按任务要求完成单元测试,如果程序有错误,报告错误,提出修改意见。

任务一：计算体积

根据输入的半径值，计算球的体积。输入数据有多组，每组占一行，每行包括一个实数，表示球的半径。输出对应的球的体积，对于每组输入数据，输出一行，计算结果保留三位小数。pi = 3.1415927;

**import** java.util.Scanner;

**public** **static** **void** func(){

Scanner sc=**new** Scanner(System.*in*);

**while**(sc.hasNext()){

**double** r=sc.nextDouble();

**double** p=3.1415927;

**double** s=4\*r\*r\*r\*p/3;

System.*out*.format("%.3f", s).println();

}

}

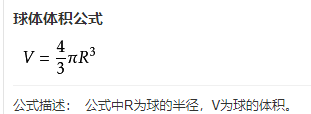
①根据本模块附件录1设计测试用例表的要求分别设计测试用例。(20分)

表 1 设计测试用例

|  |  |
| --- | --- |
| 用例要求 | 测试用例 |
| 语句覆盖 | 1 |
| 判定覆盖 | 1 |
| 条件覆盖 | 1 |
| 判定/条件覆盖 | 1 |
| 组合覆盖 | 1 |
| 路径覆盖 | 0 1 2 3 0.8（以换行符结束） |

②使用 xunit 工具和路径覆盖测试用例编写单元测试程序，展示测试结果，分析测试结果。(10分)

JUnit单元测试代码：

用例测试结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例要求 | 测试用例 | 预测结果 | 运行结果 | 结论 |
| 路径覆盖 | 0 | 0.000 | 0.000 |  |
| 1 | 4.189 | 4.189 |  |
| 2 | 33.510 | 33.510 |  |
| 3 | 113.097 | 113.097 |  |
| 0.8 | 2.145 | 2.145 |  |
|  | 0 1 2 3 0.8 | 0.000  4.189  33.510  113.097  2.145 | 0.000  4.189  33.510  113.097  2.145 |  |

分析BUG原因：

输入内容含一次换行符的运行结果与预测结果一致，输入内容以换行符结束，不能满足每组占一行的要求

下面代码存在BUG：

正确的代码应该是：

任务二：坐标求长度

输入两点坐标(x1,y1) , (x2,y2)，计算并输出两点间的距离。输入数据有多组，每组占一行，由4个实数组成，分别表示x1,y1,x2,y2,数据之间用空格隔开。例如

输入：1 3 4 6  
则输出：4.24,注意：对于每组输入数据，输出一行，结果保留两位小数。

**import** java.util.Scanner;

**public** **static** **void** func(){

Scanner in=**new** Scanner(System.*in*);

System.*out*.println("请输入第一个坐标点");

**int** x1=in.nextInt();

**int** y1=in.nextInt();

System.*out*.println("请输入第二个坐标点");

**int** x2=in.nextInt();

**int** y2=in.nextInt();

**int** distance=(**int**)Math.*sqrt*(Math.*abs*((x1-x2)\*(x1-x2))+Math.*abs*((y1-y2)\*(y1-y2)));

System.*out*.println("两点间距离是"+distance);

}

①根据本模块附件录1设计测试用例表的要求分别设计测试用例。(20分)

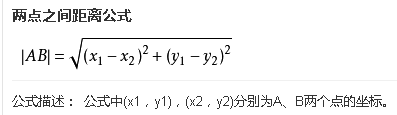


表 1 设计测试用例

|  |  |
| --- | --- |
| 用例要求 | 测试用例 |
| 语句覆盖 | 1 3 4 6 |
| 判定覆盖 | 1 3 4 6 |
| 条件覆盖 | 1 3 4 6 |
| 判定/条件覆盖 | 1 3 4 6 |
| 组合覆盖 | 1 3 4 6 |
| 路径覆盖 | 1 3 4 6  0 0 3 4  -1 -1 2 3  1 1 1 1 |

②使用 xunit 工具和路径覆盖测试用例编写单元测试程序，展示测试结果，分析测试结果。(10分)

JUnit单元测试代码：

**public** **class** LiangDianJuLiTest {

@Test

**public** **void** testFunc() {

LiangDianJuLi.*func*();

}

}

用例测试结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例要求 | 测试用例 | 预测结果 | 运行结果 | 结论 |
| 路径覆盖 | 1 3 4 6 | 4.24 | 4 | BUG |
| 0 0 3 4 | 5 | 5 |  |
| -1 -1 2 3 | 5 | 5 |  |
| 1 1 1 1 | 0 | 0 |  |

分析BUG原因：

距离为整数的没有问题，距离为非整数

下面代码存在BUG：

**int** distance=(**int**)Math.*sqrt*(Math.*abs*((x1-x2)\*(x1-x2))+Math.*abs*((y1-y2)\*(y1-y2)));

正确的代码应该是：

**double** distance=Math.*sqrt*(Math.*abs*((x1-x2)\*(x1-x2))+Math.*abs*((y1-y2)\*(y1-y2)));

System.*out*.print("两点间距离是");

System.*out*.format("%.2f",distance).println();

任务三：图形面积计算

程序实现功能如下：

定义一个抽象类sHape，它有一个计算面积的抽象方法calarea。

定义一个三角形类triangle。它有两个属性n、m，分别表示三角形的底和髙。另外，它必须继承于sHape类，并实现calarea方法来计算三角形的面积。

定义一个矩形类rectangle。它有两个属性n、m,分别表示矩形的长和宽。另外，它必须继承于sHape类，并实现calarea方法来计算矩形的面积。

定义一个圆类circle。它有一个属性n,表示圆形的半径。另外，它必须继承于sHape类，并实现 calarea 方法来计算圆形的面积。

创建一个三角形对象、一个矩形对象、一个圆形对象，然后将它们存入到一个数组。

**public** **static** **void** func(){

Shape[] s=**new** Shape[3];

s[0]=**new** Triangle(3,4);

s[1]=**new** Rectangle(5,6);

s[2]=**new** Circle(2);

System.*out*.println("三角形的面积为"+s[0].calArea());

System.*out*.println("矩形的面积为"+s[1].calArea());

System.*out*.println("圆形的面积为"+s[2].calArea());

}

**abstract** **class** Shape{

**abstract** **double** calArea();

}

**class** Rectangle **extends** Shape{

**private** **double** w,h;

**public** Rectangle(**double** w,**double** h){

**this**.h=h;

**this**.w=w;

}

@Override

**double** calArea(){

**return** w\*h;

}

}

**class** Triangle **extends** Shape{

**private** **double** a,h;

**public** Triangle(**double** a,**double** h){

**this**.a=a;

**this**.h=h;

}

**double** calArea(){

**return** a\*h/2;

}

}

**class** Circle **extends** Shape{

**private** **double** r;

**public** Circle(**double** r){

**this**.r=r;

}

**double** calArea(){

**return** 3.14\*r\*r;

}

}

①根据本模块附件录1设计测试用例表的要求分别设计测试用例。(20分)

表 1 设计测试用例

|  |  |
| --- | --- |
| 用例要求 | 测试用例 |
| 语句覆盖 | s[0]=**new** Triangle(3,4);  s[1]=**new** Rectangle(5,6);  s[2]=**new** Circle(2); |
| 判定覆盖 | s[0]=**new** Triangle(3,4);  s[1]=**new** Rectangle(5,6);  s[2]=**new** Circle(2); |
| 条件覆盖 | s[0]=**new** Triangle(3,4);  s[1]=**new** Rectangle(5,6);  s[2]=**new** Circle(2); |
| 判定/条件覆盖 | s[0]=**new** Triangle(3,4);  s[1]=**new** Rectangle(5,6);  s[2]=**new** Circle(2); |
| 组合覆盖 | s[0]=**new** Triangle(3,4);  s[1]=**new** Rectangle(5,6);  s[2]=**new** Circle(2); |
| 路径覆盖 | s[0]=**new** Triangle(3,4);  s[1]=**new** Rectangle(5,6);  s[2]=**new** Circle(2); |

② 使用 xunit 工具和路径覆盖测试用例编写单元测试程序，展示测试结果，分析测试结果。(10 分)

JUnit单元测试代码：

@Test

**public** **void** testFunc() {

// MianJi.func();

Shape[] s=**new** Shape[3];

s[0]=**new** Triangle(3,4);

s[1]=**new** Rectangle(5,6);

s[2]=**new** Circle(2);

System.*out*.println("三角形的面积为"+s[0].calArea());

System.*out*.println("矩形的面积为"+s[1].calArea());

System.*out*.println("圆形的面积为"+s[2].calArea());

}

用例测试结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例要求 | 测试用例 | 预测结果 | 运行结果 | 结论 |
| 路径覆盖 | s[0]=**new** Triangle(3,4);  s[1]=**new** Rectangle(5,6);  s[2]=**new** Circle(2); | 6.0  30.0  12.56 | 6.0  30.0  12.56 |  |
| s[0]=new Triangle(3.1,4.2);  s[1]=new Rectangle(2.5,2.6);  s[2]=new Circle(2.2); | 6.51  6.5  15.1976 | 6.51  6.5  15.1976 |  |

分析BUG原因：

运行结果与预测结果一致

下面代码存在BUG：

正确的代码应该是：

(2)作品提交要求见本模块附录2

(3)实施条件要求见本模块附录3

(4)评价标准见本模块附录4

**4.试题编号：H2-4 《“生活繁琐”计算系统》应用程序单元测试**

（1）任务描述

随着我国经济的发展,社会的进步,日交易额都在不断上升，在生活中的各种计算问题显现出来，如税收、比赛评分等问题的计算，某公司开发一套系统来辅助计算。现已经完成了相应功能代码，请按任务要求完成单元测试，如果程序有错误，报告错误，提出修改意见。

任务一：评分计算功能

计算并输出一维数组中的最大值、最小值和平均值。(如：9.8，12，45，67，23，1.98，2.55, 45)

**import** java.util.Scanner;

**public** **static** **void** func(){

Scanner sc=**new** Scanner(System.*in*);

**int** n=sc.nextInt();

**double**[] d=**new** **double**[n];

**double** max=0;//当前最大值

**double** min=0;//当前最小值

**for**(**int** i=0;i<d.length;i++){

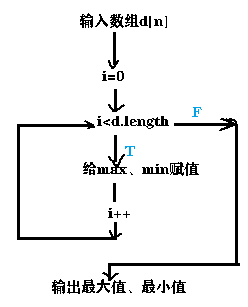
**if**(i==0)//假设第一个最大，最小

max=min=d[0];

**if**(d[i]>max)//大于最大

max=d[i];

**if**(d[i]<min)//小于最小

 min=d[i];

}

System.*out*.println("最大值："+max);

System.*out*.println("最小值："+min);

}

①根据本模块附件录 1 设计测试用例表的要求分别设计测试用例。（20 分）

表 1 设计测试用例

|  |  |
| --- | --- |
| 用例要求 | 测试用例 |
| 语句覆盖 | 1 2 3 |
| 判定覆盖 | 1 2 3 |
| 条件覆盖 | 1 2 3 |
| 判定/条件覆盖 | 1 2 3 |
| 组合覆盖 | 1 2 3 |
| 路径覆盖 | 1  1 2  1 2 3  1 3 2  3 2 1  1.1 3.3 2.2 |

②使用 xUnit 工具和路径覆盖测试用例编写单元测试程序，展示测试结果，分析测试 结果。（10 分）

JUnit单元测试代码：

@Test

**public** **void** test() {

PingFen.*func*();

}

用例测试结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例要求 | 测试用例 | 预测结果 | 运行结果 | 结论 |
| 路径覆盖 | 0 | 最大值：0.0  最小值：0.0 | 最大值：0.0  最小值：0.0 |  |
| 1 | 1 | 最大值：0.0  最小值：0.0 | Bug |
| 1 2 | 2 | 最大值：0.0  最小值：0.0 | Bug |
| 1 2 3 | 3 | 最大值：0.0  最小值：0.0 | Bug |
| 1 3 2 | 3 | 最大值：0.0  最小值：0.0 | Bug |
| 3 2 1 | 3 | 最大值：0.0  最小值：0.0 | Bug |
| 1.1 2.2 3.3 | 3.3 | 最大值：0.0  最小值：0.0 | Bug |

分析BUG原因：

输入的比零大的数字都没有正常读取，需修改赋值部分的代码

下面代码存在BUG：

**int** n=sc.nextInt();

**double**[] d=**new** **double**[n];

正确的代码应该是：

**int** n=sc.nextInt();

**double**[] d=**new** **double**[n];

**for**(**int** j=0;j<d.length;j++){

**double** m=sc.nextDouble();

d[j]=m;

}

任务二：规律数字计算

计算算式1+21+22+23+・・・+2n的值。其中 2 ≤ n ≤10。

**import** java.util.Scanner;

**public** **static** **void** func(){

**int** n,sum=1;

System.*out*.printf("please input n:\n");

Scanner sc=**new** Scanner(System.*in*);

n=sc.nextInt();

**for**(**int** i=1;i<n;++i){

sum+=20+i;

}

System.*out*.printf("sum=%d\n", sum);

}

①画出程序的控制流图。(5分)

控制流图怎么画

https://blog.csdn.net/william\_munch/article/details/85258893

<https://blog.csdn.net/qq_44654974/article/details/106179849>

**int** n,sum=1;

System.*out*.printf("please input n:\n");

Scanner sc=**new** Scanner(System.*in*);

n=sc.nextInt();

i=1;

1

2

i>=n

i<n

sum+=20+i;

3

System.*out*.printf("sum=%d\n", sum);

5

++i;

4

②根据本模块附件录 1 设计测试用例表的要求分别设计测试用例，并写出执行路径。( 15 分)

表 1 设计测试用例

|  |  |
| --- | --- |
| 用例要求 | 测试用例 |
| 语句覆盖 | 1 |
| 判定覆盖 | 1  2  10  11 |
| 条件覆盖 | 1  2  10  11 |
| 判定/条件覆盖 | 1  2  10  11 |
| 组合覆盖 | 1  2  10  11 |
| 路径覆盖 | 1  2  3  4  5  6  7  8  9  10  11 |

③使用xumt工具和路径覆盖测试用例编写单元测试程序，展示测试结果，分析测试结果(10 分)

JUnit单元测试代码：

@Test

**public** **void** testFunc() {

GuiLv.*func*();

}

用例测试结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例要求 | 测试用例 | 预测结果 | 运行结果 | 结论 |
| 路径覆盖 | 1 | sum=22 | sum=1 | BUG |
| 2 | sum=44 | sum=22 | BUG |
| 3 | sum=67 | sum=44 | BUG |
| 4 | sum=91 | sum=67 | BUG |
| 5 | sum=116 | sum=91 | BUG |
| 6 | sum=142 | sum=116 | BUG |
| 7 | sum=169 | sum=142 | BUG |
| 8 | sum=197 | sum=169 | BUG |
| 9 | sum=226 | sum=197 | BUG |
| 10 | sum=436 | sum=226 | BUG |
| 11 | n不能大于10 | sum=256 | BUG |

分析BUG原因：

小于10的运行结果比预期结果少计数一次，推断是循环体出现BUG，等于10的结果与大于10的结果不一致，退镀那程序里缺少对大于等于10的特殊处理

下面代码存在BUG：

**for**(**int** i=1;i<n;++i){

sum+=20+i;

}

正确的代码应该是：

任务三：奖金发放问题

企业发放的奖金根据利润提成。利润(i)低于或等于10万元时，奖金可提10%;利润髙于 10万元，低于 20万元时，低于 10万元的部分按10%提成，高于 10万元的部分，可可提成 7.5%; 20万到40万之间时，髙于20万元的部分，可提成 5%; 40万到60万之间时髙于40万元的部分，可提成 3%; 60万到100万之间时，髙于60万元的部分，可提成 1.5%，髙于100万元时，超过100万元的部分按1%提成，从键盘输入当月利润i。

**import** java.util.Scanner;

**public** **static** **void** func(){

Scanner sc=**new** Scanner(System.*in*);

**double** bonus1,bonus2,bonus4,bonus6,bonus10,bonus;

**int** i=sc.nextInt();

bonus1=10000\*0.1;

bonus2=bonus1+100000\*0.75;

bonus4=bonus2+200000\*0.5;

bonus6=bonus4+200000\*0.3;

bonus10=bonus6+400000\*0.15;

**if**(i<=100000)

bonus=i\*0.1;

**else** **if**(i<200000)

bonus=bonus1+(i-100000)\*0.075;

**else** **if**(i<400000)

bonus=bonus2+(i-200000)\*0.05;

**else** **if**(i<600000)

bonus=bonus4+(i-400000)\*0.03;

**else** **if**(i<1000000)

bonus=bonus6+(i-600000)\*0.015;

**else**

bonus=bonus10+(i-1000000)\*0.01;

System.*out*.println(bonus);

}

①根据本模块附件录1设计测试用例表的要求分别设计测试用例。(20分)

表 1 设计测试用例

|  |  |
| --- | --- |
| 用例要求 | 测试用例 |
| 语句覆盖 | 99999  100000  199999  399999  599999  999999  1000000  1000001 |
| 判定覆盖 | 99999  100000  199999  399999  599999  999999  1000000  1000001 |
| 条件覆盖 | 99999  100000  199999  399999  599999  999999  1000000  1000001 |
| 判定/条件覆盖 | 99999  100000  199999  399999  599999  999999  1000000  1000001 |
| 组合覆盖 | 99999  100000  199999  399999  599999  999999  1000000  1000001 |
| 路径覆盖 | 99999  100000  199999  399999  599999  999999  1000000  1000001 |

②使用 xunit 工具和路径覆盖测试用例编写单元测试程序，展示测试结果，分析测试结果。(10分)

JUnit单元测试代码：

@Test

**public** **void** testFunc() {

JiangJinFF.*func*();

}

用例测试结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例要求 | 测试用例 | 预测结果 | 运行结果 | 结论 |
| 路径覆盖 | 99999 | 9999.9 | 9999.9 |  |
| 100000 | 10000 | 10000 |  |
| 199999 | 17499.925 | 17499.925 |  |
| 399999 | 27499.95 | 94999.95 | Bug |
| 599999 | 33,499.97 | 190999.97 | Bug |
| 999999 | 39,499.985 | 250999.985 | Bug |
| 1000000 | 39500 | 305000.0 | Bug |
| 1000001 | 39500.01 | 305000.01 | Bug |

分析BUG原因：

大于20万的奖金计算都有问题，从20万以上的奖金去找问题

下面代码存在BUG：

bonus2=bonus1+100000\*0.75; //17500

bonus4=bonus2+200000\*0.5;//27500

bonus6=bonus4+200000\*0.3;//33500

bonus10=bonus6+400000\*0.15;//39500

正确的代码应该是：

bonus2=bonus1+100000\*0.075; //17500

bonus4=bonus2+200000\*0.05;//27500

bonus6=bonus4+200000\*0.03;//33500

bonus10=bonus6+400000\*0.015;//39500

（2）作品提交要求见本模块附录2

（3）实施条件要求见本模块附录3  
（4）评价标准见本模块附录4

**5.试题编号：H2-5《数值转换器系统》应用程序单元测试**

（1）任务描述  
 生活中，时时刻刻都存在着数据转换问题，如路程，时间，货币单位等，某公司开发出“数值转换”的系统，来辅助完成数据格式的转换。现已经完成了相应功能代码，请按任务  
要求完成单元测试，如果程序有错误，报告错误，提出修改意见。

任务一：小球反弹问题

一个球从n （小于100）米髙度自由落下，每次落地后反弹回原髙度的一半，再落下，再反弹。计算它在第m （小于10次）次落地时，经过的距离。

**import** java.util.Scanner;

**public** **static** **void** func(){

Scanner sc=**new** Scanner(System.*in*);

**int** sn=100;//高度100米

**int** hn=sn;//第一个反弹高度

**int** i=0;

**int** num=0;

num=sc.nextInt();

**for**(i=2;i<=num;i++){

sn=sn+hn\*2;//第i次落地经过多少米

hn=hn/2;//第i次反弹的高度

}

System.*out*.println("第"+i+"次落地经过"+sn+"米");

System.*out*.println("第"+i+"次反弹"+hn+"米");

}

①根据本模块附件录1设计测试用例表的要求分别设计测试用例。（20分）

表 1 设计测试用例

|  |  |
| --- | --- |
| 用例要求 | 测试用例 |
| 语句覆盖 | 2 |
| 判定覆盖 | 2  10 |
| 条件覆盖 | 2  10 |
| 判定/条件覆盖 | 2  10 |
| 组合覆盖 | 2  10 |
| 路径覆盖 | 1  2  3  4  5  6  7  8  9  10 |

②使用 xunit 工具和路径覆盖测试用例编写单元测试程序，展示测试结果，分析测试结果。（10分）

JUnit单元测试代码：

用例测试结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例要求 | 测试用例 | 预测结果 | 运行结果 | 结论 |
| 路径覆盖 | 1 | 第1次落地经过100米  第1次反弹50米 | 第2次落地经过100米  第2次反弹100米 | BUG |
| 2 | 第2次落地经过200米  第2次反弹25米 | 第3次落地经过300米  第3次反弹50米 |  |
| 3 | 第3次落地经过250米  第3次反弹12.5米 | 第4次落地经过400米  第4次反弹25米 |  |
| 4 | 第4次落地经过275米  第4次反弹6.25米 |  |  |
| 5 | 第5次落地经过287.5米  第5次反弹3.125米 |  |  |
| 6 | 第6次落地经过293.75米  第6次反弹1.5625米 |  |  |
| 7 | 第7次落地经过296.875米  第7次反弹0.78125米 |  |  |
| 8 | 第8次落地经过298.4375米  第8次反弹0.390625米 |  |  |
| 9 | 第9次落地经过299.21875米  第9次反弹0.1953125米 |  |  |
| 10 | 请输入小于10的值 |  |  |

分析BUG原因：

第1、2次反弹运行结果与预期结果不一致，，推断初始设置有Bug.

下面代码存在BUG：

正确的代码应该是：

**public** **static** **void** func(){

Scanner sc=**new** Scanner(System.*in*);

**double** sn=100;//高度100米

**double** hn=sn/2;//第一个反弹高度

**int** i=1;

**int** num=0;

num=sc.nextInt();

**if**(num==1){

System.*out*.println("第"+i+"次落地经过"+sn+"米");

System.*out*.println("第"+i+"次反弹"+hn+"米");

}**else**{

**for**(i=1;i<num;i++){

sn=sn+hn\*2;//第i次落地经过多少米

hn=hn/2;//第i次反弹的高度

}

System.*out*.println("第"+i+"次落地经过"+sn+"米");

System.*out*.println("第"+i+"次反弹"+hn+"米");

}

}

任务二：停电停多久问题

lee 的老家住在工业区，日耗电量非常大。

今年7月，传来了不幸的消息，政府要在7、8月对该区进行拉闸限电。政府决定从7月 1 日起停电，然后隔一天到 7月 3 日再停电，再隔两天到 7月 6 日停电，依次下去，每次都比上一次长一天。lee 想知道自己到家后到底要经历多少天倒霉的停电。请编写程序帮他算一算。根据放假日期、开学日期，日期限定在7、8月份，且开学日期大于放假日期，计算出停电天数。

**import** java.util.Scanner;

**public** **static** **void** func(){

**int** month,day;

**int** n=1,s1=1,num=0;

**int** m,s2;

Scanner sc=**new** Scanner(System.*in*);

System.*out*.print("请输入放假月");

month=sc.nextInt();

System.*out*.print("请输入放假日：");

day=sc.nextInt();

System.*out*.print("请输入开学月");

**int** month2=sc.nextInt();

System.*out*.print("请输入开学日");

**int** day2=sc.nextInt();

**if**(month>month2){

System.*out*.println("放假日期不能超过开学日期");

**return**;

}

**switch**(month2){

**case** 7:num=day2;

**case** 8:num=day2+31;**break**;

**default**:**break**;

}

**while**(s1<num){

s1+=n;

n++;

}

m=1;s2=1;

**while**(s2<=(month-7)\*31+day){

s2+=m;

m++;

}//剩下的天数

System.*out*.println(n-m);

}

①根据本模块附件录1设计测试用例表的要求分别设计测试用例。(20分)

表 1 设计测试用例

|  |  |
| --- | --- |
| 用例要求 | 测试用例 |
| 语句覆盖 | 8 1 7 20  7 5 8 20  7 1 7 31  7 2 9 1 |
| 判定覆盖 | 8 1 7 20  7 5 8 20  7 1 7 31  7 2 9 1 |
| 条件覆盖 | 8 1 7 20  7 5 8 20  7 1 7 31  7 2 9 1 |
| 判定/条件覆盖 | 8 1 7 20  7 5 8 20  7 1 7 31  7 2 9 1 |
| 组合覆盖 | 8 1 7 20  7 5 8 20  7 1 7 31  7 2 9 1 |
| 路径覆盖 | 8 1 7 20  7 15 7 1  7 1 7 1  7 1 7 2  7 1 7 3  ……  7 1 7 21  7 1 7 28  7 1 8 1  7 1 8 5  ……  7 1 8 30  7 1 8 31  8 5 8 5  8 1 8 5  ……  8 1 8 30  8 1 8 31  7 5 8 20  7 10 7 31  7 2 9 1 |

②使用 xunit 工具和路径覆盖测试用例编写单元测试程序，展示测试结果，分析测试结果。(10分)

JUnit单元测试代码：

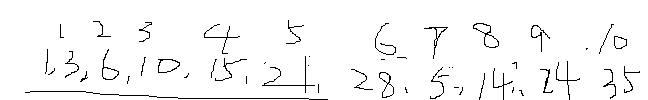
@Test

**public** **void** testFunc() {

TingDian.*func*();

}

用例测试结果：



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例要求 | 测试用例 | 预测结果 | 运行结果 | 结论 |
| 路径覆盖 | 8 1 7 20 | 放假日期不能超过开学日期 | 放假日期不能超过开学日期 |  |
| 7 15 7 1 | 放假日期不能超过开学日期 | 3 |  |
| 7 1 7 1 | 1 | 7 | BUG |
| 7 1 7 2 | 1 | 7 | BUG |
| 7 1 7 3 | 2 | 7 | BUG |
| …… |  |  |  |
| 7 1 7 21 | 6 | 9 |  |
| 7 1 7 28 | 7 | 10 | BUG |
| 7 1 8 1 | 7 | 7 |  |
| 7 1 8 5 | 7 | 7 |  |
| …… |  |  |  |
| 7 1 8 24 | 10 | 10 |  |
| 7 1 8 31 | 10 | 10 |  |
| 8 5 8 5 | 0 | 0 |  |
| 8 5 8 13 | 1 | 1 |  |
| …… |  |  |  |
| 8 1 8 30 |  |  |  |
| 8 1 8 31 |  |  |  |
| 7 5 8 20 |  |  |  |
|  | 7 10 7 31 |  |  |  |
| 7 2 9 1 | 月份必须是在7、8月份 | -2 | BUG |

分析BUG原因：

放假日期不能超过开学日期的判定只针对不同月份的日期，没有针对同月份的日期；还有7月份的停电天数计算错误，计算停电天数的程序有bug

下面代码存在BUG：

正确的代码应该是：

任务三：筛选奇数问题

根据输入正整数s，从低位开始取出s中的奇数位上的数，依次构成一个新数t，髙位仍放在髙位，低位仍放在低位，返回to例如，当s中的数为7654321时，t中的数为7531。

**import** java.util.Scanner;

**public** **static** **void** func(){

Scanner sc=**new** Scanner(System.*in*);

**int** s=0,t=0;

**int** n=1,i=0,num=1;

s=sc.nextInt();

**while**(**true**){

**if**(s==0)**break**;

n=s%10;

**if**(n%2!=0){

t=t+n\*num;

i++;

num=1;

**for**(**int** j=0;j<=i;j++)

num\*=10;

}

s/=10;

}

System.*out*.println(t);//21

}

①根据本模块附件录1设计测试用例表的要求分别设计测试用例。(20分)

表 1 设计测试用例

|  |  |
| --- | --- |
| 用例要求 | 测试用例 |
| 语句覆盖 | 0  7654321 |
| 判定覆盖 | 0  7654321 |
| 条件覆盖 | 0  7654321 |
| 判定/条件覆盖 | 0  7654321 |
| 组合覆盖 | 0  7654321 |
| 路径覆盖 | 0  1  21  12  13  123  222  153  …  7654321 |

②使用 xunit 工具和路径覆盖测试用例编写单元测试程序，展示测试结果，分析测试结果。(10分)

JUnit单元测试代码：

用例测试结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例要求 | 测试用例 | 预测结果 | 运行结果 | 结论 |
| 路径覆盖 | 0 | 0 | 0 |  |
| 1 | 1 | 1 |  |
| 2 | 0 | 0 |  |
| 21 | 1 | 1 |  |
| 12 | 1 | 1 |  |
| 13 | 13 | 103 | BUG |
| 123 | 13 | 103 | BUG |
| 222 | 0 | 0 |  |
| 153 | 153 | 1503 | BUG |
| … | … | … | … |
| 7654321 | 7531 | 75301 | BUG |

分析BUG原因：

超过2个奇数的运行结果与预期结果不一致，推断是第二次奇数取十位数出现BUG

下面代码存在BUG：

**for**(**int** j=0;j<=i;j++)

num\*=10;

正确的代码应该是：

**for**(**int** j=1;j<=i;j++)

num\*=10;

（2）作品提交要求见本模块附录 2

（3）实施条件要求见本模块附录 3

（4）评价标准见本模块附录 4

**6.试题编号：H2-6《智能统计系统》应用程序单元测试**

（1）任务描述

生活中在处理各种问题时都存在数据统计的问题，例如学生人数的统计，火车买票人数的统计等，某公司开发出一套数据统计系统，用来辅助完成数据的统计功能。现已经完成了相应功能代码，请按任务要求完成单元测试，如果程序有错误，报告错误，提出修改意见。

任务一：统计今天是该年的第几天问题

根据输入一个年份值和一个月份值，输出该月的天数。 （说明：一年有1 2个月，大月的天数是31，小月的天数是30。 2月的天数比较特殊，遇到闰年是29天，否则为28天。例女如，输入2011、3，则输出31天。）

**import** java.util.Scanner;

**public** **static** **void** func(){

System.*out*.print("请输入您要判断的年份：");

Scanner sc=**new** Scanner(System.*in*);

**int** y=sc.nextInt();

System.*out*.print("请输入你要判断的月份为：");

**int** m=sc.nextInt();

**int** d=0;

**switch**(m){

**case** 1:

d=31;**break**;

**case** 2:

d=28;**break**;

**case** 3:

d=31;**break**;

**case** 4:

d=30;**break**;

**case** 5:

d=31;**break**;

**case** 6:

d=30;**break**;

**case** 7:

d=31;**break**;

**case** 8:

d=31;**break**;

**case** 9:

d=30;**break**;

**case** 10:

d=31;

**case** 11:

d=30;**break**;

**case** 12:

d=31;**break**;

**default**:

System.*out*.println("对不起，您输入的月份有误！");

**break**;

}

**if**(y%400==0||y%4==0||y%100!=0){

System.*out*.println(y+"是闰年");

**if**(m==2)

++d;

System.*out*.println("请输入该月份的天数为："+d);

}**else**{

System.*out*.println(y+"不是闰年");

System.*out*.println("您输入该月份的天数为："+d);

}

}

①根据本模块附件录1设计测试用例表的要求分别设计测试用例。（20分）

表 1 设计测试用例

|  |  |
| --- | --- |
| 用例要求 | 测试用例 |
| 语句覆盖 | 2021 1  2021 2  2021 3  2021 4  2021 5  2021 6  2021 7  2021 8  2021 9  2021 10  2021 11  2021 12  2020 2 |
| 判定覆盖 | 2021 1  2021 2  2021 3  2021 4  2021 5  2021 6  2021 7  2021 8  2021 9  2021 10  2021 11  2021 12  2020 2 |
| 条件覆盖 | 2021 1  2021 2  2021 3  2021 4  2021 5  2021 6  2021 7  2021 8  2021 9  2021 10  2021 11  2021 12  2020 2  2000 2  2100 2 |
| 判定/条件覆盖 | 2021 1  2021 2  2021 3  2021 4  2021 5  2021 6  2021 7  2021 8  2021 9  2021 10  2021 11  2021 12  2020 2  2000 2  2100 2 |
| 组合覆盖 | 2021 1  2021 2  2021 3  2021 4  2021 5  2021 6  2021 7  2021 8  2021 9  2021 10  2021 11  2021 12  2020 2  2000 2  2100 2 |
| 路径覆盖 | 2021 1  2021 2  2021 3  2021 4  2021 5  2021 6  2021 7  2021 8  2021 9  2021 10  2021 11  2021 12  2020 2  2000 2  2100 2 |

②使用 xunit 工具和路径覆盖测试用例编写单元测试程序，展示测试结果，分析测试结果。（10分）

JUnit单元测试代码：

用例测试结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例要求 | 测试用例 | 预测结果 | 运行结果 | 结论 |
| 路径覆盖 | 2021 1 |  |  |  |
| 2021 2 | 2021不是闰年  请输入该月份的天数为：28 | 2021是闰年  请输入该月份的天数为：29 | Bug |
| 2021 3 | 2021不是闰年  请输入该月份的天数为：31 | 2021是闰年  请输入该月份的天数为：31 | Bug |
| 2021 4 |  |  |  |
| 2021 5 |  |  |  |
| 2021 6 |  |  |  |
| 2021 7 |  |  |  |
| 2021 8 |  |  |  |
| 2021 9 |  |  |  |
| 2021 10 | 2021不是闰年  请输入该月份的天数为：31 | 2021是闰年  请输入该月份的天数为：30 | Bug |
| 2021 11 | 2021不是闰年  请输入该月份的天数为：30 | 2021是闰年  请输入该月份的天数为：30 | Bug |
| 2021 12 |  |  |  |
| 2020 2 | 2020是闰年  请输入该月份的天数为：29 | 2020是闰年  请输入该月份的天数为：29 |  |
| 2000 2 | 2000是闰年  请输入该月份的天数为：29 | 2000是闰年  请输入该月份的天数为：29 |  |
| 2100 2 | 2100不是闰年  请输入该月份的天数为：28 | 2100是闰年  请输入该月份的天数为：29 | Bug |

分析BUG原因：

闰年的判断有BUG，10月份的判断有BUG

下面代码存在BUG：

**case** 10:

d=31;

**case** 11:

d=30;**break**;

**if**(y%400==0||y%4==0||y%100!=0){

System.*out*.println(y+"是闰年");

正确的代码应该是：

任务二：统计纸片对折问题

假设一张足够大的纸，纸张的厚度为 0.5 毫米。请问对折多少次以后，可以达到某个高度（如：8844.43米）。请编写程序输出对折次数。

**public** **static** **void** func(){

**double** h=8844.43\*1000;//珠穆朗玛峰的高度为8844.43米

**double** p=0.5;//一张纸一层的厚度为0.5毫米

**int** sum=0;

**for**(**int** a=0;a<=8844;a++){

**if**(p<h){

p\*=2;//每折叠一次增加一倍高度

sum+=1;//折叠一次

}

**if**(p>=h)**break**;

}

System.*out*.println(sum);

}

①根据本模块附件录1设计测试用例表的要求分别设计测试用例。(20分)

表 1 设计测试用例

|  |  |
| --- | --- |
| 用例要求 | 测试用例 |
| 语句覆盖 | h=4 |
| 判定覆盖 | h=4 |
| 条件覆盖 | h=4 |
| 判定/条件覆盖 | h=4 |
| 组合覆盖 | h=4 |
| 路径覆盖 | h=1  h=2  h=4  h=8  h=16  …  h=8388608  h=16777216  h=8844430 |

②使用 xunit 工具和路径覆盖测试用例编写单元测试程序，展示测试结果，分析测试结果。(10分)

JUnit单元测试代码：

@Test

**public** **void** testFunc() {

**double** h=1;//8844.43\*1000;//珠穆朗玛峰的高度为8844.43米

**double** p=0.5;//一张纸一层的厚度为0.5毫米

**int** sum=0;

**for**(**int** a=0;a<=8844;a++){

**if**(p<h){

p\*=2;//每折叠一次增加一倍高度

sum+=1;//折叠一次

}

**if**(p>=h)**break**;

}

System.*out*.println(sum);

}

用例测试结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例要求 | 测试用例 | 预测结果 | 运行结果 | 结论 |
| 路径覆盖 | h=1 | 1 |  |  |
| h=2 | 2 |  |  |
| h=4 | 3 |  |  |
| h=8 | 4 |  |  |
| h=16 | 5 |  |  |
| … |  |  |  |
| h=8388608 | 24 |  |  |
| h=16777216 | 25 |  |  |
| h=8844430 | 24 | 25 | BUG |

分析BUG原因：

等于2的24次方输出没有问题，大于2的24运算结果与预测不符合

下面代码存在BUG：

**if**(p<h){

p\*=2;//每折叠一次增加一倍高度

sum+=1;//折叠一次

}

正确的代码应该是：

任务三：统计同构数问题

判断m~n之间的同构数。同构数是指这个数为该数平方的尾数，例如5的平方为25,6的平方为36, 25的平方为625,则5、6、25都为同构数。其中m>1,n<99。

**public** **static** **void** func(){

**int** s=0,num;

**for**(**int** i=2;i<=99;i++){

**if**(i>10){

s=i\*i;

s=s%100;

**if**(i==s)

System.*out*.println(i);

}

s=i\*i;

s=s%10;

**if**(i==s)

System.*out*.println(i);

}

}

①根据本模块附件录1设计测试用例表的要求分别设计测试用例。(20分)

表 1 设计测试用例

|  |  |
| --- | --- |
| 用例要求 | 测试用例 |
| 语句覆盖 | i=25  i=5 |
| 判定覆盖 | i=25  i=5 |
| 条件覆盖 | i=25  i=5  i=7  i=11 |
| 判定/条件覆盖 | i=25  i=5  i=7  i=11 |
| 组合覆盖 | i=25  i=5  i=7  i=11 |
| 路径覆盖 | i=1  i=2  i=3  …  i=98  i=99  i=5  i=6  i=25  i=76 |

②使用 xunit 工具和路径覆盖测试用例编写单元测试程序，展示测试结果，分析测试结果。(10分)

JUnit单元测试代码：

@Test

**public** **void** testFunc() {

**int** s=0,num;

**for**(**int** i=2;i<=99;i++){

**if**(i>10){

s=i\*i;

s=s%100;

**if**(i==s){

System.*out*.print(i);

System.*out*.println(i\*i);}

}

s=i\*i;

s=s%10;

**if**(i==s){

System.*out*.print(i);

System.*out*.println(i\*i);}

}

}

用例测试结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例要求 | 测试用例 | 预测结果 | 运行结果 | 结论 |
| 路径覆盖 | i=2 | 空白 | 空白 |  |
| i=3 | 空白 | 空白 |  |
| … |  |  |  |
| i=98 | 空白 | 空白 |  |
| i=99 | 空白 | 空白 |  |
| i=5 | 525 |  |  |
| i=6 | 636 |  |  |
| i=25 | 25625 |  |  |
| i=76 | 765776 |  |  |

分析BUG原因：

无BUG

下面代码存在BUG：

正确的代码应该是：

（2）作品提交要求见本模块附录 2

（3）实施条件要求见本模块附录 3

（4）评价标准见本模块附录 4

**7.试题编号：H2-7《中国结图案打印系统》应用程序单元测试**

( 1 )任务描述

中国结是一种中国特有的手工编织工艺品,它所显示的情致与智慧是中华古老文明的一个写照。原本是由旧石器时代的缝衣打结,后推展至汉朝的仪礼记事,再演变成今日的装饰手艺。当代多用来装饰室内、亲友间的馈赠礼物及个人的随身饰物。因为其外观对称精致,可以代表中国悠久的历史，符合中国传统装饰的习俗和审美观念，故命名为中国结。现在a公司想要开发一个中国结图案打印系统。已经完成了相应功能代码,请按任务要求完成单元测试，如果程序有错误，报告错误，提出修改意见。

任务一：主结长度问题

公司现在需要打印中国结的主结(位于中间，最大的那一个结)，为了打印出漂亮新颖的主结，于是设计打印主结的长度满足可以被 7 整除这个条件。现在公司需要统计某个范围内能被7整除的整数的个数，以及这些能被7整除的数的和。根据输入一个整数n,输出1〜n之间能被7整除的整数的个数，以及这些能被7整除的数的和。

**import** java.util.Scanner;

**public** **static** **void** func(){

Scanner sc=**new** Scanner(System.*in*);

**int** n=sc.nextInt();

**if**(2<=n&&n<=1000){

**int** count=0;

**int** sum=0;

**for**(**int** i=1;i<n;i++){

**if**(i/7==0){

count++;

sum+=i;

}

}

System.*out*.println(count);

System.*out*.println(sum);

}

}

①根据本模块附件录1设计测试用例表的要求分别设计测试用例。(20分)

表 1 设计测试用例

|  |  |
| --- | --- |
| 用例要求 | 测试用例 |
| 语句覆盖 | n=7 |
| 判定覆盖 | n=7  n=1  n=1001 |
| 条件覆盖 | n=7  n=1  n=2  n=1001  n=1000 |
| 判定/条件覆盖 | n=7  n=1  n=2  n=1001  n=1000 |
| 组合覆盖 | n=7  n=1  n=2  n=1001  n=1000 |
| 路径覆盖 | n=1  n=2  n=3  n=6  n=7  n=8  n=13  n=14  n=15  ...  n=993  n=994  n=995  n=999  n=1000  n=1001 |

②使用 xunit 工具和路径覆盖测试用例编写单元测试程序，展示测试结果，分析测试结果。(10分)

JUnit单元测试代码：

@Test

**public** **void** testFunc() {

ZhongGuoJie.*func*();

}

用例测试结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例要求 | 测试用例 | 预测结果 | 运行结果 | 结论 |
| 路径覆盖 | n=1 | 空白 | 空白 |  |
| n=2 | 0 0 | 1 1 | BUG |
| n=3 | 0 0 | 2 3 | BUG |
| n=6 | 0 0 | 0 0 | BUG |
| n=7 | 1 7 | 0 0 | BUG |
| n=8 | 1 7 |  | BUG |
| n=13 | 1 7 |  | BUG |
| n=14 | 2 21 |  | BUG |
| n=15 | 2 21 |  | BUG |
| ... |  |  | BUG |
| n=993 | 141 70077 |  | BUG |
| n=994 | 142 71071 |  | BUG |
| n=995 | 142 71071 |  | BUG |
| n=999 | 142 71071 |  | BUG |
| n=1000 | 142 71071 |  | BUG |
| n=1001 | 空白 | 空白 |  |

分析BUG原因：

不是7的倍数也都计算进去了，推断是判断7的倍数代码有BUG

下面代码存在BUG：

**for**(**int** i=1;i<n;i++){

**if**(i/7==0){

正确的代码应该是：

**for**(**int** i=1;i<=n;i++){

**if**(i%7==0){

任务二：副结长度问题

公司设计的中国结还需要副结(主结周围的结)，于是打算设计副结的长度满足是素数这个条件。现在公司需要统计出某个范围内哪些数是素数。根据输入的整数n,输出1〜n之间的素数。

**import** java.util.Scanner;

**public** **static** **void** func(){

Scanner sc=**new** Scanner(System.*in*);

System.*out*.println("请输入一个数");

**long** n=sc.nextLong();

**boolean** flag=**true**;

**for**(**long** i=2;i<=n;i++){

flag=**true**;

**for**(**long** j=2;j<=i;j++){

**if**(i%j==0)

flag=**false**;

}

**if**(flag)System.*out*.println(i);

}

}

①根据本模块附件录1设计测试用例表的要求分别设计测试用例。(20分)

表 1 设计测试用例

|  |  |
| --- | --- |
| 用例要求 | 测试用例 |
| 语句覆盖 | 5 |
| 判定覆盖 | 2  1 |
| 条件覆盖 | 1  2 |
| 判定/条件覆盖 | 1  2 |
| 组合覆盖 | 1  2 |
| 路径覆盖 | 1  2  3  4  5 |

② 使用 xunit 工具和路径覆盖测试用例编写单元测试程序，展示测试结果，分析测试结果。（10 分）

JUnit单元测试代码：

用例测试结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例要求 | 测试用例 | 预测结果 | 运行结果 | 结论 |
| 路径覆盖 | 1 | 空白 | 空白 |  |
| 2 | 2 | 空白 |  |
| 3 | 2  3 | 空白 |  |
| 4 | 2  3 | 空白 |  |
|  | 5 | 2  3  5 | 空白 |  |

分析BUG原因：

所有输入的数据都没有输出，原因是flag=false语句每次都执行了，推断是判断素数的条件有BUG

下面代码存在BUG：

**for**(**long** j=2;j<=i;j++){

**if**(i%j==0)

flag=**false**;

}

正确的代码应该是：

**for**(**long** j=2;j<i;j++){

**if**(i%j==0)

flag=**false**;

}

任务三：打印中国结图案  
由于中国结的形状是菱形图案，所以现在公司需要设计一个打印菱形的方法。根据输入的整数n,打印出有n\*2-l行的菱形。

例如输入整数 4，则屏幕输出如下菱形。

**import** java.util.Scanner;

**public** **static** **void** func(){

**int** number;

Scanner sc=**new** Scanner(System.*in*);

number=sc.nextInt();

**if**(number<0){

System.*out*.println("input is error!");

**return**;

}

**for**(**int** a=1;a<number;a++){

**for**(**int** b=number-a;b>0;b--){

String m=" ";

System.*out*.print(m);

}

**for**(**int** c=1;c<a;c++){

String n="\*";

System.*out*.print(n);

}

System.*out*.print(" ");

}

**for**(**int** a=1;a<=number;a++){

**for**(**int** b=0;b<a;b++){

String m=" ";

System.*out*.print(m);

}

**for**(**int** c=number-a;c>0;c--){

String n="\*";

System.*out*.print(n);

}

System.*out*.println("");

}

}

①根据本模块附件录1设计测试用例表的要求分别设计测试用例。(20分)

表 1 设计测试用例

|  |  |
| --- | --- |
| 用例要求 | 测试用例 |
| 语句覆盖 | -1  3 |
| 判定覆盖 | -1  1  3 |
| 条件覆盖 | -1  1  3 |
| 判定/条件覆盖 | -1  1  3 |
| 组合覆盖 | -1  1  3 |
| 路径覆盖 | -1  0  1  2  3  4  5 |

② 使用 xunit 工具和路径覆盖测试用例编写单元测试程序，展示测试结果，分析测试结果。（10 分）

JUnit单元测试代码：

@Test

**public** **void** testFunc() {

ZhongGuoJiePrint.func();

用例测试结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例要求 | 测试用例 | 预测结果 | 运行结果 | 结论 |
| 路径覆盖 | -1 | input is error! | input is error! |  |
| 0 | 空白 | 空白 |  |
| 1 | \* | 空白 | BUG |
| 2 | \*  \* \*  \* | \* | BUG |
| 3 | \*  \* \*  \* \* \*  \* \*  \* | \* \*\*  \* | BUG |
| 4 |  |  |  |
| 5 |  |  |  |

分析BUG原因：

输入1没有输出\*，推断是输出\*的程序代码有BUG

下面代码存在BUG：

**for**(**int** a=1;a<number;a++){

**for**(**int** b=number-a;b>0;b--){

正确的代码应该是：

**int** number;

Scanner sc=**new** Scanner(System.*in*);

number=sc.nextInt();

**if**(number<0){

System.*out*.println("input is error!");

**return**;

}

**for**(**int** a=1;a<=number;a++){

**for**(**int** b=number-a;b>=0;b--){

String m=" ";

System.*out*.print(m);

}

**for**(**int** c=1;c<a;c++){

String n="\*";

System.*out*.print(n);

System.*out*.print(" ");

}

System.*out*.println();

}

**for**(**int** a=1;a<=number;a++){

**for**(**int** b=1;b<a;b++){

String m=" ";

System.*out*.print(m);

}

**for**(**int** c=number-a;c>=0;c--){

String n="\*";

System.*out*.print(n);

System.*out*.print(" ");

}

System.*out*.println("");

}