Assignment2

1. 猫爬树问题

题目描述

一只擅长爬树的猫在树干上每跳起来一次可以往上爬`1`dm，`2`dm或者`3`dm。对于一棵高度为`n`dm的树，它从地面开始爬，跳若干次之后可以爬到树顶（假设猫的体力是无限的）。那么，对于这棵高度为`n`dm的树，猫有多少种不同的方法可以从地面爬到树顶？

例如，当树的高度为`3`dm, 猫有以下4种不同的方法可以选择：

- 跳三次，每次跳`1`dm。
- 先跳`1`dm，再跳`2`dm。
- 先跳`2`dm，再跳`1`dm。
- 直接一下跳`3`dm。

在单个文件`main.cpp`中实现该程序。

输入输出范围与格式要求

- 输入n是一个正整数，它的范围是$0 < n <= 50$。
- 输出一个正整数，即爬到树顶方法的总数量。

示例

示例一

输入

```
3
```

输出

```
4
```

示例二

输入

```
4
```

输出

```
7
```

Hint

- 在课堂上，我们了解了完成同一种功能的递归程序在效率上有多大差异。此题目也需要你思考分析如何能够减少递归计算的次数，从而减少被使用的栈帧、优化程序运行时间。
- 对于0 < n < = 50，可能的输出数据在什么范围内？

2. 浮点数转换为整数

题目描述

在编程解决问题时，我们有时需要将浮点数按照不同的方式转换为整数。

- 将小数部分直接舍去，只保留整数部分。C++中的显式类型转换运算符就是这样工作的。
- 四舍五入，使浮点数转换为与之最接近的整数。
- 向绝对值更大的方向取整。

为这三种转换方法分别设计函数`Truncate`、`Round`和`LargerAbs`。它们读入一个`float`类型的浮点数，返回转换之后的整数。 使用命令`t`、`r`和`l`表示需要使用哪种转换方式。命令`t`表示舍去小数，`r`表示四舍五入，`l`表示向绝对值更大的方向取整。

在单个文件`main.cpp`中实现该程序。

输入输出范围与格式要求

- 输入包含一个字符和一个待转换的`float`类型浮点数。字符是决定以何种方式进行转换的命令。输入的浮点数范围不会超过`int`类型。
- 输出转换后的整数即可。如果输入的命令字符不属于`t`、`r`和`l`之中的任何一种，输出`Wrong Command`。

示例

示例一

输入

```
t 3.987
```

输出

```
3
```

示例二

输入

```
r -2.78
```

输出

```
-3
```

3. 迭代逼近计算平方根

题目描述

在计算一个浮点数`x`的平方根时，可以采用迭代逼近的方法。这种方法的主要思想是：

- 先猜测一个初始答案`g = x/2`。
- 易知，真正的答案，即`x`的平方根，就在`g`和`x/g`之间。猜测新的答案为`g`和`x/g`的平均值，使之成为新的`g`。
- 判断该答案`g`是否符合精度要求。在该题目中，我们使用`eps`表示所需精度。如果新生成的`g`与上一轮旧的`g`之间的误差不超过`eps`，即可认为该答案已满足精度要求。否则跳转到第二步，继续迭代。

例如，对于浮点数`2.3`，我们设置精度`eps = 0.1`。

- 第一轮：`g = 1.15, x/g = 2`
- 第二轮：`g = 1.575, x/g = 1.460...`，与第一轮的`g`相比误差大于精度要求。

- 第三轮：`g = 1.518...`，与第二轮的`g`相比误差符合精度要求。认为该答案即为`2.3`的平方根。

编写函数`MySqrt`，读入一个浮点数`x`和要求的精度`eps`，用迭代逼近的方法得到它的平方根并返回。

在单个文件`main.cpp`中实现该程序。

输入输出范围与格式要求

- 输入包含：一个大于零的`float`类型的浮点数`x`、大于零的`float`类型的精确度`eps`。`eps`的小数位数不会超过六位。
- 输出它足够精确的平方根`g`。输出结果保留三位小数。

示例

示例一

输入

```
2.3 0.1
```

输出

```
1.518
```

示例二

输入

```
0.34 0.001
```

输出

```
0.583
```

Hint

- 输出的小数位数控制可以使用Assignment1中介绍过的`<iomanip>`头文件中的函数`fixed`和`setprecision`。

4. 数字的性质

题目描述

- 毕达哥拉斯曾说：“6象征着完满的婚姻以及健康和美丽，因为它的部分是完整的，并且其和等于自身。” **完美数**是一类特殊的自然数。它所有的真因子（即除了自身以外的约数）的和，恰好等于它本身。例如，6的真因子有1、2、3，它们加起来就等于6本身。
- **素数**是指除了1和它本身以外不再有其他因数的自然数。例如，17是一个素数，它的因子只有1和它本身。

我们需要实现自己的C++库，来得到判断一个自然数`n`是否为完美数或素数的接口。

- 在文件`NumberProperty.h`中定义函数接口`isPerfect`和`isPrime`，分别用来判断`n`是否是完美数或素数。如果是就返回`1`，不是就返回`0`。
- 在文件`NumberProperty.cpp`中实现接口中的函数。
- 在文件`main.cpp`中，程序读入自然数`n`，输出两个布尔值，它们中间有空格，分别用来表示是否是完美数、是否是素数。

输入输出范围与格式要求

- 输入一个自然数*n*，它的范围是*0 < = n < 2^31*
- 输出包含两个空格隔开的布尔值，分别表示*n*是否是完美数、是否是素数。

示例

示例一

输入

```
6
```

输出

```
1 0
```

示例二

输入

```
7
```

输出

```
0 1
```

5. 我的数学库

题目描述

在2、3和4三道题目中，我们分别实现了浮点数转换为整数的多种方式、求浮点数平方根和判断一个自然数是否是完美数或素数的程序。现在，我们可以把它们分别整合到不同的文件中，并把函数接口抽象出来，从而能够在主函数中进行调用。

- 在文件`Convert2Int.h`和`Convert2Int.cpp`中分别声明和定义题目2中的函数`Truncate`、`Round`和`LargerAbs`
- 在文件`Sqrt.h`和`Sqrt.cpp`中分别声明和定义题目3中的函数`MySqrt`。
- 将题目4中的`NumberProperty.h`和`NumberProperty.cpp`添加到该项目中。
- 在文件`MyMath.h`中，包含这些库的头文件。
- 在`main.cpp`中包含头文件`MyMath.h`。在主函数中输入一个浮点数，程序会先求出它精确度为`eps = 0.1`的平方根，然后按照四舍五入得到它的近似整数，并判断这个整数是否是素数。这些结果会打印到标准输出流。

输入输出范围与格式要求

- 输入是一个不为负的`float`类型浮点数。
- 输出包含三行：
  - `The square root is:` + 该数精确度为`eps = 0.1`的平方根，输出保留一位小数。
  - `Its nearest interger is:` + 四舍五入得到的它的近似整数。
  - `Whether the interger is a prime number:` + 0或1。

示例

示例一

输入

```
2.3
```

输出

```
The square root is:1.5
Its nearest interger is:2
Whether the interger is a prime number:1
```

示例二

输入

```
36
```

输出

```
The square root is:6.0
Its nearest interger is:6
Whether the interger is a prime number:0
```

提交文件结构

```
<your student number>.zip
|- 1_cat
| |- main.cpp
|
|- 2_convert
| |- main.cpp
|
|- 3_sqrt
| |- main.cpp
|
|- 4_property
| |- main.cpp
| |- NumberProperty.h
| |- NumberProperty.cpp
|
|- 5_math
| |- main.cpp
| |- NumberProperty.h
| |- NumberProperty.cpp
| |- Convert2Int.h
| |- Convert2Int.cpp
| |- Sqrt.h
| |- Sqrt.cpp
| |- MyMath.h
```

`judger`和`judger_batch`的用法与第一次作业类似。例如，如果你想批量测试第一题，你在当前文件夹中新建`1_cat/main.cpp`，那么批量测试的命令即为：

```
python judger_batch.py -S 1_cat -I data/1_cat -O data/1_cat -T 1_cat
```