Assignment 1 C++ 基础

1. 数位求和

题目描述

本题需要你实现一个程序，对于给定的正整数 `n`,求出 `n` 在各个数位上的数字之和。

输入输出格式及示例

输入和输出均为单个数字，例如：

输入

```
119
```

输出

```
11
```

输入

```
2023
```

输出

```
7
```

数据范围

`0 < n < 1000000000`

2. 寻找较大数

> 在本题中，我们将尝试最简单的流处理，尝试在连续输入的一系列数中找出前两个较大的数字。

题目描述

本题中需要你实现一个程序，该程序读取一系列的整数直到用户输入`0`作为终止输入的标志。当`0`被输入时，你的程序应该输出在此之前读取到的所有数字中最大的两个。

输入输出格式

输入为一系列（至少2个）用空格分隔开的整数，最后以终止符0结尾。

```
223 251 317 636 766 607 607 0
```

输出2个整数，同样用空格分隔开。

```
766 636
```

数据范围

对于所有的输入`n`，`-100000000 < n < 100000000`

3. 3n+1问题

> 1979年，印第安纳大学认知科学教授道格拉斯-侯世达（Douglas Hofstadter）写了《哥德尔、艾舍尔、巴赫》，他将其描述为 "以刘易斯-卡罗尔的精神对思想和机器进行的隐喻赋格"。该书获得了普利策文学奖，多年来已成为计算机科学的经典之一。它的大部分魅力来自于它所包含的数学怪事和谜题，其中许多可以用计算机程序的形式来表达。
>
> 本书中提及的其中一个问题被称为考拉兹猜想（Collatz conjecture），又称为3n+1问题。对于任意正整数n，我们根据以下规则执行：
>
> 1. 若n=1,则计算终止
> 2. 若n为偶数，则令n = n/2
> 3. 若n为1以外的奇数，令 n = 3n+1
>
> 考拉兹猜想的内容即为对任意正整数应用此规则，都能够最终使得n=1。
>
> 例如，当n=6时，该计算经历的步骤为6,3,10,5,16,8,4,2,1。共经过8步计算使得n=1。

本题需要你实现一个用于测试3n+1问题的程序，给定输入正整数`n`，对其执行上述的计算规则直到终止，并输出完成计算所需的步骤数。

输入输出格式

输入

```
4
```

输出

```
2
```

输入

```
6
```

输出

```
8
```

数据范围

`0<n<1000`

4. π近似计算

德国数学家莱布尼茨（1646-1716）发现了对常数π的近似计算公式：

$$\frac{\pi}{4} \simeq 1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}+\ldots$$

随着计算的$\frac{1}{2n-1}$项数越来越多，通过这个公式对于π的计算得到的结果就越精确。本题需要你在C++中实现上述的计算公式来估计π的值，根据输入中给定的计算项数`n`来输出对π相应精度的估计值。

为了保证结果的一致性，请在计算的过程中采用`double`类型作为每一步浮点数运算的类型。输出估计值时统一采用9位小数，请在程序开始处引入`iomanip`头文件：

```
#include <iomanip>
```

并在输出时使用`fixed`和`setprecision`函数来选择格式，例如：

```
cout << fixed << setprecision(9) << pi;
```

输入输出格式

输入

```
2
```

输出

```
2.666666667
```

输入

```
500
```

输出

```
3.139592656
```

数据范围

`0 < n < 10001`

5. 素因数分解

> 任何正整数都能够被分解为一系列素数的乘积，这样的因数分解结果是唯一的，被称为素因数分解。例如60可以被分解为`60 = 2 * 2 * 3 * 5`。同一个素数可能在素因数分解的序列中出现多次。

本题需要你实现一个素因数分解的程序，给定正整数`n > 1`，将`n`的素因数分解序列按从小到大的顺序输出，以空格分割。

输入输出格式

输入

```
6
```

输出

```
2 3
```

输入

```
60
```

输出

```
2 2 3 5
```

数据范围

`1 < n < 100000000`

提交格式

你提交的文件结构应该类似如下形式：

```
<your student number>.zip
|- 1_digit_sum
|  |- main.cpp
|
|- 2_top2_big_number
|  |- main.cpp
|
|- 3_3n+1
|  |- main.cpp
|
|- 4_pi_approximation
|  |- main.cpp
|
|- 5_prime_factorization
|  |- main.cpp
```

Tips

Judger的基本使用方法参见canvas `文件` 中提供的样例和使用演示，本次作业的提交格式和测试点位置与样例略有不同，在 `data` 文件夹中包含了本次题目的测试点。我们建议你在当前文件夹中为每一题按照上述的提交格式要求创建一个文件夹，并将源代码 `main.cpp` 放在创建的文件夹内（即用codeblocks建立名为 `1_digit_sum` 的工程）。

在这样的配置下，使用judger的示例为：

```
 python judger.py -I data/1_digit_sum/1.in -O data/1_digit_sum/1.out -S
1_digit_sum -T 1_digit_sum
```

在编写完全部程序并测试完毕后，将5个文件夹中除了main.cpp以外的文件删除之后一起打包压缩，最后用自己的学号命名，即可提交。