Assignment 1

本次实验主要练习C++基础语法。

本次实验共分为5题。

对应知识点

第一题：热身

第二题：C++的实际运用、分支结构

第三题：整数类型数据范围

第四题：循环结构

第五题：综合运用

要求

输入输出严格按照题目表述中的格式，不要输出额外的信息。

参考`codeing style.pdf`文件，规范编程。另外在程序中写上必要的注释。coding style会占一小部分的分数。

自测

我们每道题目都给出了一些测试点便于同学自测。同学可以自行使用任意文本编辑器打开.in和对应的.out文件查看。

另外，为了避免同学与助教的测试环境不同造成的问题，请同学提交作业前使用`judger`进行测试。

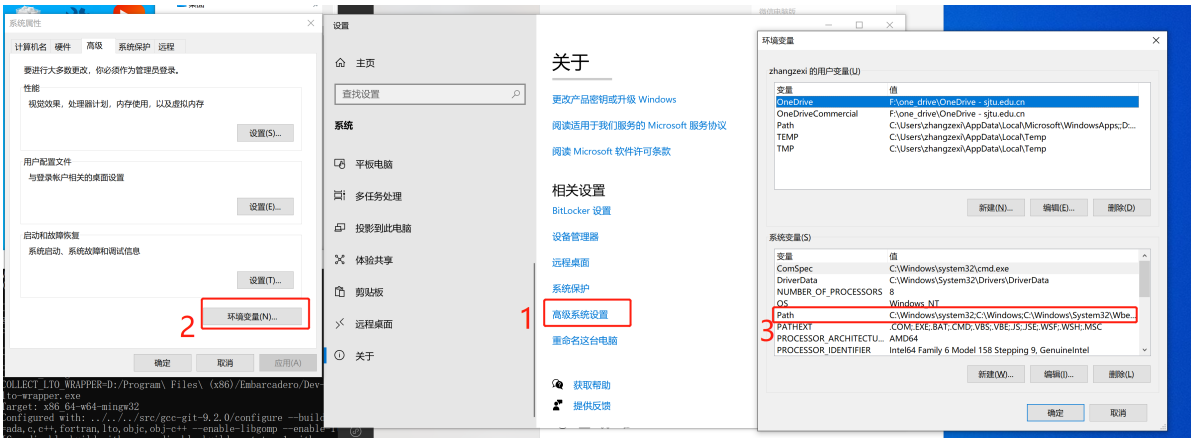
配置g++到环境变量。

可以先打开Windows powershell或cmd，输入`g++ -v`命令，如果提示"不是内部或外部命令也不是可运行的程序"，则说明`g++`未被添加到环境变量中。

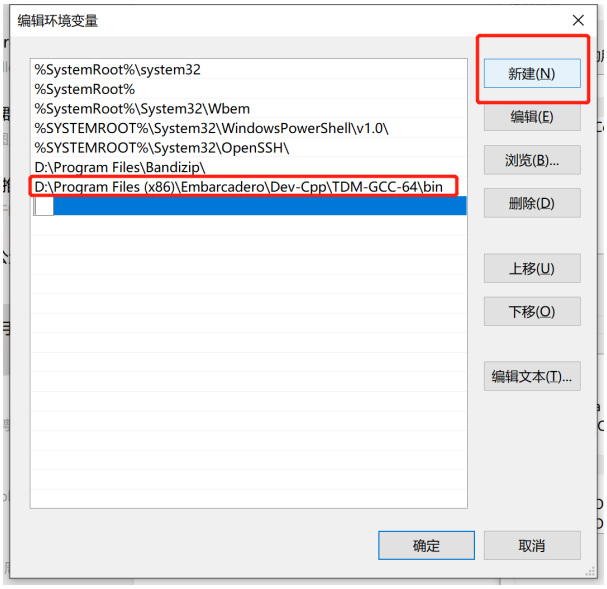
添加`g++`到环境变量方法：

①找到`g++.exe`的安装位置(与安装codeblocks时的设置有关）。

②打开环境变量设置。



③将第①步中的`g++.exe`所在文件夹目录（通常是`bin`文件夹）增加到Path中，如下图所示。



④配置完成后，打开一个**新的**powershell或cmd窗口，运行 `g++ -v` ，如果出现可以显示g++的版本信息则配置完成。

```
C:\Users\zhangzexi>g++ -v
Using built-in specs.
COLLECT_GCC=g++
COLLECT_LTO_WRAPPER=D:/Program\ Files\ (x86)/Embarcadero/Dev-Cpp/TDM-GCC-64/bin/../libexec/gcc/x86_64-w64-mingw32/9.2.0/
lto-wrapper.exe
Target: x86_64-w64-mingw32
Configured with: ../../../src/gcc-git-9.2.0/configure --build=x86_64-w64-mingw32 --enable-targets=all --enable-languages
=ada,c,c++,fortran,lto,objc,obj-c++ --enable-libgomp --enable-lto --enable-graphite --enable-cxx-flags=-DWINPTHREAD_STAT
IC --disable-build-with-cxx --disable-build-poststage1-with-cxx --enable-libstdcxx-debug --enable-threads=posix --enable
-version-specific-runtime-libs --enable-fully-dynamic-string --enable-libstdcxx-threads --enable-libstdcxx-time --with-g
nu-ld --disable-werror --disable-nls --disable-win32-registry --enable-large-address-aware --disable-rpath --disable-sym
vers --prefix=/mingw64tdm --with-local-prefix=/mingw64tdm --with-pkgversion=tdm64-1 --with-bugurl=http://tdm-gcc.tdragon
.net/bugs
Thread model: posix
gcc version 9.2.0 (tdm64-1)
```

Tips

将`judger_batch.py`和`judger.py`和`data`放在作业工程文件夹的同一目录下，每一个文件夹是你的工程文件夹，有一个`main.cpp`以及其他的工程文件。

```
|- 1_fizzBuzz
|
|- 2_gpa
|
|- 3_encrypt
|
|- 4_getZero
|
|- 5_uglyNumber
|
|- data
|
|- judger.py
|
|- judger_batch.py
```

确保你的电脑安装有Python和Cpp，并且按照将g++添加到环境变量。在这样的配置下，cd到该目录，使用judger的示例为：

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

批量版judger使用说明

由于每个题目提供了5个测试点，手动逐个测试稍显繁琐，我们在通用judger之外提供批量版的judger。批量版judger可以一次性测试一个题目的5个测试点，使用时将原来的`输入文件`和`标准输出文件`改为`输入路径`和`标准输出路径`。使用新judger的示例为：

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

该命令将一次性测试`data/1_fizzBuzz`路径下从`1.in,1.out`到`5.in,5.out`的全部测试点。

Assignment 1 C++ 基础

1.FizzBuzz

题目描述

从1到n打印数字。对于3的倍数，打印“Fizz”而不是数字，对于5的倍数，打印“Buzz”。对于既是3的倍数又是5的倍数的数字，打印“FizzBuzz”。

输入输出格式及示例

输入为单个整数，输出为“Fizz”、“Buzz”、“FizzBuzz”和单个整数，中间有空格。

例如：

输入

```
15
```

输出

```
1 2 Fizz 4 Buzz Fizz 7 8 Fizz Buzz 11 Fizz 13 14 FizzBuzz
```

输入

```
19
```

输出

```
1 2 Fizz 4 Buzz Fizz 7 8 Fizz Buzz 11 Fizz 13 14 FizzBuzz 16 17 Fizz 19
```

数据范围

`0 < n < 1000000`

2. 计算GPA

题目描述

根据交大的计算规则，根据课程的成绩和学分，计算平均积点GPA。

百分制	等级制	积点	评价的参考标准
[95—100]	A+	4.3	课程考核通过，对知识的理解、掌握、运用情况综合评价优秀。
[90—95)	A	4.0	
[85—90)	A-	3.7	
[80—85)	B+	3.3	课程考核通过，对知识的理解、掌握、运用情况综合评价一般。
[75—80)	B	3.0	
[70—75)	B-	2.7	
[67—70)	C+	2.3	课程考核通过，对知识的理解、掌握、运用情况综合评价基本合格。
[65—67)	C	2.0	
[62—65)	C-	1.7	
[60—62)	D	1.0	
⩾60	P(Pass)	N/A	合格
<60	F(Failure)	0	不及格、未通过
DF(Deferred Final Examination)	DF(Deferred Final Examination)	N/A	缓考

`假设IEEE班的孩子们既聪明又努力，没有挂科的。`

$$\text{平均积点GPA} = \frac{\sum(\text{课程学分} \times \text{积点})}{\sum \text{课程学分}}$$

输入输出格式

输入一个数n，代表门数。

接下来依次输入课程的考试成绩和学分，中间通过空格隔开。

```
2
90 6
96 2
```

输出

```
4.075000
```

【注意】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6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(6) << totalGPA << endl;
```

数据范围

对于所有的输入`n`，`0 < n < 100`

3. 密码学初探

```
文本加密是一种通过转换原始文本使其在未经授权的情况下难以理解的技术。历史上，人们使用了许多不同的加密方法，从简单的替换字母到复杂的数学算法，以保护敏感信息不被未经授权的人读取。以下是一些历史上常见的文本加密方法和一些加密历史事件：

1.凯撒密码（Caesar Cipher）：由古罗马军事领袖凯撒使用的一种简单的替换密码，将字母按照一定的规律替换成另外的字母。例如，将每个字母向后移动三位。
2.维吉尼亚密码（Vigenère Cipher）：16世纪法国外交家布拉谢尔·德维吉尼亚发明的一种多表密码，通过使用不同的凯撒密码表格来加密消息。
3.恩尼格玛（Enigma）：二战期间纳粹德国使用的复杂机械加密设备，它使用了多个可互换的转轮来进行替换加密。
4.RSA算法：由罗纳德·李维斯特（Ron Rivest）、阿迪·萨莫尔（Adi Shamir）和伦纳德·阿德曼（Leonard Adleman）在1977年提出的一种公钥加密算法，基于大素数分解的难题。
5.现代对称加密算法：如AES（高级加密标准）和DES（数据加密标准），它们使用相同的密钥进行加密和解密。
6.现代非对称加密算法：如RSA算法、椭圆曲线密码算法（ECC），使用公钥加密，私钥解密。
7.量子密码：基于量子力学原理的一种加密方法，利用量子态的性质来保护信息安全。

这些加密方法的发展历程反映了人类对信息安全的不断追求，从最早的简单替换到复杂的数学算法，加密技术在保护通信和数据安全方面发挥着关键作用。
```

编写一个加密程序，输入12位学号，将其加密后输出。加密方法是将该数每一位的数字加3，超过10从0开始。比如1得到4，9得到2。

【提示】int能表示的最大整数为2^31-1。超过范围后，应该考虑采用其他的数据类型。

输入输出格式

输入

```
523030910204
```

输出

```
856363243537
```

数据范围

`12位整数`

4. 得到 0 的操作数

给你两个**非负**整数 num1 和 num2 。

每一步 **操作** 中，如果 `num1 >= num2` ，你必须用 `num1 减 num2`；否则，你必须用`num2 减 num1`。
返回使`num1 = 0`或`num2 = 0`的操作数 。

```
例如，num1 = 5 且 num2 = 4 ，应该用 num1 减 num2 ，因此，得到 num1 = 1 和 num2 = 4 。然而，如果 num1 = 4且 num2 = 5 ，一步操作后，得到 num1 = 4 和 num2 = 1 。
```

输入输出格式

输入

```
2 3
```

输出

```
3
```

输入

```
10 10
```

输出

```
1
```

数据范围

`0 <= num1, num2 <= 10^5`

5. 丑数

丑数就是只包含质因数 2、3 和 5 的正整数。

输入输出格式

给定一个整数 n ，请你判断 n 是否为 丑数 。如果是，返回 `Yes`；否则，返回 `No`。

输入

```
6
```

输出

```
Yes
```

输入

```
11
```

输出

```
No
```

数据范围

`int`的范围

提交格式

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。

你提交的文件结构应该和以下形式**完全一样**：

```
<your student number>.zip
|- 1_fizzBuzz
|  |- main.cpp
|
|- 2_gpa
|  |- main.cpp
|
|- 3_encrypt
|  |- main.cpp
|
|- 4_getZero
|  |- main.cpp
|
|- 5_uglyNumber
|  |- main.cpp
```