B卷第一题-求圆形的周长和面积

题目 | #635

题目描述

编写程序，定义一个圆形的类，求圆形的周长和面积，规定 $\pi = 3.14$。

输入

1个正整数，代表圆形的半径r。

输出

圆形的周长L，面积S，均是double类型，以空格分隔，行尾无空格

样例输入

```
3
```

样例输出

```
18.84 28.26
```

注意

必须用类实现周长和面积的计算函数，否则计0分。

数据保证计算结果在 $double$ 表示范围内。

题目描述

某水果店为了促销，制定了如下的定价策略，P代表某种水果的单价，W代表购买某种水果的重量： 苹果：打八折，需支付$P * W * 0.8$元；香蕉：半价，需支付$P * W/2$元；橘子：如果$W >= 10$,则打半价，即需支付$W * P/2$元；如果$W >= 5$,则打七五折，则需支付$W * P * 0.75$元，其它情况不打折。

请你编写一个抽象基类$Fruit$，包括2个保护成员，分别是$Price$（单价，int类型）、$Weight$（重量，int类型）。定义纯虚函数$SumMoney$，用于计算顾客购买的水果总价。以$Fruit$作为基类，派生出$Apple$、$Banana$、$Orange$类，实现计算顾客购买水果总价的功能。

输入

多行，每行格式为$C\ W\ P$，中间用空格分隔，分别代表水果类型（a、b、o分别代表苹果、香蕉、橘子），W（int型）、P（int型）分别代表顾客购买的相应水果的重量和单价。输入字符q结束输入。

输出

购买的水果的总价。

样例输入

```
a 3 9
o 5 9
b 15 5
q
```

样例输出

```
92.85
```

注意

必须实现抽象基类、多态、动态联编、否则计0分。

题目描述

假设你有一个简单的单向链表，链表的节点包含一个整数值（大于等于0，且小于10）。现在，你需要重载+运算符，使得两个链表可以进行一种特殊的相乘操作：反转相乘具体规则如下：

1. 首先将两个链表分别反转。
2. 然后从头节点开始，对应位置的节点值相乘。
3. 如果链表长度不等，较短的链表后面的缺失值视为1。
4. 如果相乘的结果大于等于10，则进位到下一个节点继续参与计算（如果不存在则创建一个新节点）。

例如，链表A为$5\ 2\ 3$，链表B为$4\ 5\ 6$，反转后分别为$3\ 2\ 5$和$6\ 5\ 4$，相乘后的链表为$8\ 1\ 1\ 2$（因为$3*6=18$, 进位1，$2*5+1=11$, 进位1, $5*4+1=21$, 进位2）。

要求

1. 定义链表节点结构体$ListNode$，包含整数值val和指向下一个节点的指针$next$。
2. 定义链表类$LinkedList$，需要重载$+$运算符，使其接受两个链表头节点作为参数，并返回反转相乘的链表头节点。
3. 再$LinkedList$类中编写辅助函数用于反转链表以及输入、输出函数。
4. 如果A或者B为空链表，则返回另一链表的反转链表；如果A和B同为空链表，则返回$NULL$。

输入

第一行为链表A，

第二行为链表B，

其中空链表用$NULL$表示。

输出

反转相乘后的结果

样例输入：

输出

反转相乘后的结果

样例输入：


```
5 2 3
4 5 6
```

样例输出：

```
8 1 1 2
```

题目描述： 请编写实现一个通用的$MyStack$(栈，后进先出$LIFO$)模板类，该模板类实现了一个能够存储常见数据类型（int,$char$）的栈，栈最大容量$MAXLEN$为20。该模板类形式及主要成员函数如下：

```
template<class T>
class MyStack{
...
MyStack(T array[], int len);    //构造函数，根据数组构造
void push( const &T value);     //入栈
T pop();                                //出栈
bool empty();                   //空栈判断
int count();                    //栈中成员数
void show();                    //打印栈，打印栈中所有元素和元素个数，空格分隔
void clear();                   //清空栈
};
```

完成该模板类及成员函数之后，请在$main$()函数中验证，验证要求和方式如下：

1、定义两个数组，整数类型数组和字符类型数组，分别初始化为固定数值，然后利用这两个数组进行构造类对象，数组如下格式：

```
int iarray[] = {2,3,5,7,11,13,17,19,23,29,31,37};   //整数数组，初始数据固定
char carray[] = "this_is_a_teststring";           //字符数组，初始数据固定
```

2、从标准输入端读取操作标记符和操作数，操作符和操作标记如下（第一个字符为操作标记符，后面的x为操作数）：

输入 操作要求 说明：

d 展示栈，栈列输出“$None$” 执行$show$()方法打印栈数据和元素个数

s 终止操作此栈 无输出信息

i x 让操作数x入栈 无输出信息，若栈满则输出“$FULL$”

s 终止操作此栈 无输出信息

i x 让操作数x入栈 无输出信息，若栈满则输出“$FULL$”

o x 从栈中出队x个元素，x为int类型 无输出信息，若处理到空栈则输出“$None$”

3、栈有容量限制$MAXLEN$，因此，在栈已满时，入栈将无法进行，直接输出“$FULL$”。同样，当栈为空时，再执行出栈操作时栈将无数据，直接输出1次“$None$”。

程序输入

程序输入为两行，第一行为对int类型栈的操作序列，第二行为对$char$类型栈的操作序列，每一行均以s结尾（字符栈的入栈字符不会出现' '空格字符）。

程序输出

依次为输入操作码对应的输出信息，用空格分隔。