自定义函数练习题

选自《一本通》

【题目描述】

求正整数 2 和 n 之间的完全数（一行一个数）。

完全数：因子之和等于它本身的自然数，如 6=1+2+3

【输入】

输入 n(n ≤ 5000)。

【输出】

一行一个数，按由小到大的顺序。

【输入样例】　　7

【输出样例】　　6

5.0-1

```
#include<iostream>
using namespace std;
int k(int x)
{
	int j,s=0;
	for(j=1;j<=x-1;j++)// 枚举 1 到 x 中，判断是否为 x 的因子
	{
		if(x%j==0) s=s+j;// 若是 x 的因子，就由 s 累加
	}
	return s;
}
int main()
{
	int i,n;
	cin>>n;
	for(i=2;i<=n;i++)
	{
		if(k(i)==i)
		cout<<i;
	}
	return 0;
}
```

【题目描述】

编程求 2-n(n 为大于 2 的正整数）中有多少个素数。

【输入】

输入 n(2 ≤ n ≤ 50000)。

【输出】素数个数。

【输入样例】 【输出样例】

10 4

5.0-2

```
#include<iostream>
#include<cmath>
using namespace std;
int sushu(int i)//sushu 素数，求素数函数
{
	bool f=false;
	int x=2;
	while(x<=floor(sqrt(i))&&i%x!=0) //floor 四舍五入取整函数
	{	x++;  }

	if(x>floor(sqrt(i)))		f=true;
	return f;
}
int main()
{
	int i,n,s=0;
	cin>>n;
	for(i=2;i<=n;i++)
	{
		if(sushu(i)==true)
		{	s++;
			//cout<<i<<" ";
		}
	}
	cout<<endl<<s;
	return 0;
}
```

【题目描述】

已知：

$$m = \frac{\max(a,b,c)}{\max(a+b,b,c)\times\max(a,b,b+c)}$$

输入 a,b,c，求 m。把求三个数的最大数 max(x,y,z) 分别定义成函数和过程来做。

【输入】

输入 a,b,c。

【输出】

求 m，保留到小数点后三位。

【输入样例】

1 2 3

【输出样例】

0.200

5.0-3

```
#include<iostream>
#include<cstdio>
using namespace std;
double MAX(double a,double b,double c)
{
	return max(a,max(b,c))/(max(a+b,max(b,c))*max(a,max(b,b+c)))*1.0;
}
int main()
{
	double a,b,c;
	cin>>a>>b>>c;
	cout<<MAX(a,b,c)<<endl;
	printf("%.3f",MAX(a,b,c));
	return 0;
}
```

【题目描述】

如果一个自然数是素数，且它的数字位置经过对换后仍为素数，则称为绝对素数，例如13。试求出所有二位绝对素数。

【输入】

（无）

【输出】

所有二位绝对素数（由小到大，一个数一行）。

5.0-4

```
#include<iostream>
#include<cmath>
using namespace std;
int sushu(int i)
{
	bool f=false;
	int x=2;
	while(x<=floor(sqrt(i))&&i%x!=0)
	{
		x++;
	}
	if(x>floor(sqrt(i))) f=true;
	return f;
}
int main()
{
	int i;
	for(i=10;i<=99;i++)
	{
		if(sushu(i)==true&&sushu((i%10)*10+(i/10))==true)
		cout<<i<<" ";

	}
	return 0;
}
```

【题目描述】

自然数a的因子是指能整除a的所有自然数，但不含a本身。例如12的因子为：1,2,3,4,6。若自然数a的因子之和为b，而且b的因子之和又等于a，则称a,b为一对“亲和数” 。求最小的一对亲和数(a<>b)。

【输入】

（无）

【输出】

1行，分别为a和b(a<b)。

```
220 284
```

5.0-5

```
#include<iostream>//5. 亲和数
using namespace std;
int num(int n)
{
	int x=0;
	for(int i=1;i<=n/2;i++)
	{
		if(n%i==0) x+=i;
	}
	return x;
}
int main()
{
	for(int i=1;;i++)// 因设置找到第一个亲和数立即退出，所以没有限制到哪个数字
	{
		if(num(num(i))==i&&i!=num(i))
		{
			cout<<i<<" "<<num(i)<<endl;
			break;
		}
	}
	return 0;
}
```

【题目描述】

如果一个数从左边读和从右边读都是同一个数，就称为回文数。例如 6886 就是一个回文数，求出所有的既是回文数又是素数的三位数。

【输入】（无）

【输出】

所有的既是回文数又是素数的三位数。一个数一行。

```
101 131 151 181 191 313 353
373 383 727 757 787 797 919
929
```

5.0-6

```
#include<iostream>//6. 回文数
#include<cmath>
using namespace std;
int sushu(int i)
{
	bool f=false;
	int x=2;
	while(x<=floor(sqrt(i))&&i%x!=0)
	{
		x++;
	}
	if(x>floor(sqrt(i))) f=true;
	return f;
}
int main()
{
	int i;
	for(i=100;i<=999;i++)
	{
		if(sushu(i)==true&&i/100==i%10) cout<<i<<" ";
	}
	return 0;
}
```

根据公式：

$$arctanx(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots 和 \pi = 6arctanx(\frac{1}{\sqrt{3}})$$

定义函数 $arctanx(x)$，求当最后一项小于10^{-6}时π的值。

【输出】

π 的值。保留到小数点后 10 位。

5.0-7

```
#include<iostream>
#include<cmath>
#include<cstdio>
using namespace std;
double ax(double x)
{
	int i=1;
	double s=0.0,y,t=x;
	while(abs(t/s)>=1e-6)
	{
		s+=t/i;
		t=-1*x*x*t;
		i+=2;
	}
	s+=t/i;
	return s;
}
int main()
{
	double a=ax(1/sqrt(3));
	double pi=6*a;
	printf("%0.10f",pi);
	return 0;
}
```

```
3.1415926345
```

【题目描述】哥德巴赫猜想的命题之一是：大于 6 的偶数等于两个素数之和。编程将 6～100 所有偶数表示成两个素数之和。

【输出】分行输出：例如：6=3+3　　8=3+5　　...（每个数只拆开一次，保证第一个加数最小）

5.0-8

```
#include<iostream>//8. 哥德巴赫猜想      重点学习
#include<cmath>
using namespace std;
int n,x;
bool ss(int i)
{
	int x=2;
	while(x<=floor(sqrt(i))&&i%x!=0)
	{
		x++;
	}
	if(x>floor(sqrt(i))) return true;
	else return false;
}
int main()
{
	for(int x=6;x<=100;x+=2)
	{
		for(int i=2;i<=x/2;i++)
		{
			if(ss(i)&&ss(x-i))//(ss(i) 为 true, 并且 ss(x-i) 为 true
			{
				cout<<x<<"="<<i<<"+"<<x-i<<endl;
				break;
			}
		}
	}
	return 0;
}
```

函数的递归调用

选自《一本通》

【题目描述】

用递归的方法求 1+2+3+……+N 的值。

【输入】

输入 N。

【输出】

输出和。

【输入样例】

5

【输出样例】

15

5.3-1

```
#include<iostream>
using namespace std;
int nm(int n)// 计算 1+2+……+n 的值
{
	if(n==0) return 0;
	else return n+nm(n-1);
}
int x;
int main()
{
	cin>>x;
	cout<<nm(x);
	return 0;
}
```

【题目描述】

用递归函数输出斐波那契数列第 n 项。0,1，1，2，3，5，8，13……

【输入】

一个正整数 n，表示第 n 项。

【输出】

第 n 项是多少。

【输入样例】

3

【输出样例】

1

5.3-2

```
#include<iostream>
using namespace std;
int nm(int n)// 输出斐波那契数列第 n 项
{
	if(n==1) return 0;// 判断是否到达递归边界：n=1
	else if(n==2) return 1;// 判断是否到达递归边界：n=2
	else return nm(n-1)+nm(n-2);// 否则继续递归
}
int x;
int main()
{
	cin>>x;
	cout<<nm(x);
	return 0;
}
```

【题目描述】

输入一个非负整数，输出这个数的倒序数。例如输入 123，输出 321。

【输入】

输入一个非负整数（保证个位不为零）。

【输出】

输出倒序的数。

【输入样例】

123

【输出样例】

321

5.3-3

```
#include<iostream>
using namespace std;
void daoxu(int x)// 输出 x 的倒序数
{
	cout<<x%10;// 输出 x 的最后一位
	if(x>=10) daoxu(x/10);// 判断是否到达递归边界 x<10，否则继续递归
}
int main()
{
	int n;
	cin>>n;
	daoxu(n);
	return 0;
}
```

【题目描述】

用递归算法将一个十进制数 X 转换成任意进制数 M（M ≤ 16）。

【输入】

一行两个数，第一个十进制数 X，第二个为进制 M。

【输出】

输出结果。

【输入样例】

31 16 {将十进制 31 转化为十六进制数}

【输出样例】

1F

5.3-4// 本题选自《一本通》

```
#include<iostream>// 十进制转 16 进制之内
using namespace std;
char d[16]={'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};
int x,y;
int zhuanhuan(int n,int k)// 十进制数字 n 转 k 进制数字
{
	int r;
	r=n%k;n=n/k;
	if(n!=0) zhuanhuan(n,k);
	cout<<d[r];
}
int main()
{
	cin>>x>>y;
	zhuanhuan(x,y);
	return 0;
}
```

```
100 16
64


30 16
1E

1024 2
10000000000
```

自定义函数练习题

选自《编程竞赛宝典》

浮点数求最大值

输入三个浮点值，求其中的最大值，请使用函数编程解决。

输入

输入三个浮点值。

输出

输出其中的最大值。

样例

输入

1.1 2.2 3.3

输出

3.3

6.1-1// 求最大值

```
#include <bits/stdc++.h>
using namespace std;

float Max(float a,float b,float c)
{
  return max(max(a,b),c);
}

int main()
{
  float x,y,z;
  cin>>x>>y>>z;
  cout<<Max(x,y,z)<<endl;
  return 0;
}
```

判断平方数

写一个判断平方数的函数，判断输入的整数 x 是否为平方数并输出结果。

输入　有多组数据，每组一行，为一个整数 x。

输出　每组输出一行，如果 x 是平方数输出 1，否则输出 0。

样例

输入

25

99

输出

1

0

6.1-2-a

```
#include <bits/stdc++.h>// 判断平方数
using namespace std;
int x;
void Square(int n);       // 对子函数的声明
int main()
{
      while(cin>>x)
      Square(x);
      return 0;
}

void Square(int n)        //void 表示函数无返回值
{
      for(int i=1; i<n; i++)
      {
            if(i*i==n)
            {
                  cout<<1;     break;
            }
            if(i==n-1)         cout<<0;
      }
}
```

```
6.1-2-b// 判断平方数
#include <bits/stdc++.h>
using namespace std;

void Square(int n);       // 对子函数的声明

int main()
{
  int x;
  while(cin>>x)
    Square(x);
  return 0;
}

void Square(int n)        //void 表示函数无返回值
{
  for(int i=1; n>0; i+=2)// 由 1+1，1+3，1+3+5... 为平方数推导而来
    n-=i;
  cout<<(n==0?1:0)<<endl;// 直接输出结果，所以无 return 值
}
```

哥德巴赫猜想

输入整数 a 和 b，试验证 a~b 区间内的所有正偶数都能够分解为两个素数之和（即验证哥德巴赫猜想对 a~b 以内的正偶数成立）。

输入　　两个整数 a，b（2 ＜ a ＜ b\1e500 000，b － a ＜ 200 000）a，b（2 ＜ a ＜ b ≤ 500000，b － a ＜ 200000）。

输出　　输出 a~b 区间内的正偶数的素数之和，每个占一行，例如 4 ＝ 2 ＋ 2，如果有多种可能，只输出一种，即第一个素数最小的。

样例

输入 3 6

输出 4=2+2　　6=3+3

```
6.1-3-a// 哥德巴赫猜想
#include<bits/stdc++.h>
using namespace std;

int fflag(int i)              // 判断是否为素数
{
  if(i==2)
    return 1;
  if(!(i%2))                  // 如果是偶数，return 0
    return 0;
  int k=sqrt(i);
  for(int j=3; j<=k; j+=2)
    if(!(i%j))
      return 0;
  return 1;                   // 如果是素数，return 1
}

int main()
{
  for(int i=4; i<=2000; i+=2)
    for(int n=2; n<i; n++) // 将偶数 i 分解为两个整数
      if(fflag(n) && fflag(i-n))      // 分别判断两个整数是否均为素数
      {
        printf("%d=%d+%d\n",i,n,i-n);// 若均是素数则输出
        break;
      }
  return 0;
}
```

```
6.1-3-b// 哥德巴赫猜想—筛选法预处理
#include<bits/stdc++.h>
using namespace std;
bool p[10005];                              //保存素数

void GetPrime()                             //预处理出所有素数
{
  for(int i=2; i<=2000; i++)
    p[i]=true;
  int m=sqrt(2000);
  for(int i=2; i<=m; i++)
    if(p[i])
      for(int j=i*i; j<=2000; j+=i)
        p[j]=false;
}

int main()
{
  GetPrime();
  for(int i=4; i<=2000; i+=2)
    for(int n=2; n<i; n++)                  //将偶数 i 分解为两个整数
      if(p[n] && p[i-n])
      {
        printf("%d=%d+%d\n",i,n,i-n); //若均是素数则输出
        break;
      }
  return 0;
}
```

6.1-3-c

```
#include<bits/stdc++.h>// 哥德巴赫猜想—筛选法预处理
using namespace std;
bool p[500005]; // 保存素数
int a,b,i,j,y,n;
void GetPrime(int y) // 预处理出所有素数
{
  for(i=2; i<=y; i++) p[i]=true;// 初始化所有数字为真（假设所有数字为质数）

  for(int i=2; i<=sqrt(y); i++)
    if(p[i])// 如果某数为质数（为真）
      for(j=i*i; j<=y; j+=i)//j 是 i 的倍数，判断为合数
        p[j]=false;
}
int main()
{
  cin>>a>>b;
  GetPrime(b);
  for(i=2; i<=b; i++)
  {
      if(p[i]) cout<<i<<" ";// 观察筛选出的质数
  }

  for(i=a; i<=b; i+=2)
    for(n=2; n<i; n++)              // 将偶数 i 分解为两个整数
      if(p[n]&&p[i-n])
      {
            cout<<endl<<i<<"="<<n<<"+"<<i-n<<endl; // 若均是素数则输出
        break;// 发现一个符合条件的等式即退出本轮循环
      }
  return 0;
}
```

106004 - 约分

$\frac{3}{6}$的约分形式应该是$\frac{1}{2}$，试编程输入n个分数，输出其对应的约分形式。

输入

输入第一行为一个整数$n(n \leq 500000)$，表示有n个分数。随后n行，每行为一个分子和分母，分子与分母之间以“/”间隔。

输出

输出n行，分别对应其约分形式的分数，分子与分母之间以“/”间隔。

样例

输入	复制
1 3/9	
输出	复制
1/3	

函数的递归调用

选自《编程竞赛宝典》

有 5 个人坐在一起，问第 5 个人多少岁，他说比第 4 个人大 2 岁；问第 4 个人多少岁，他说比第 3 个人大 2 岁；问第 3 个人多少岁，他说比第 2 个人大 2 岁； 问第 2 个人多少岁，他说比第 1 个人大 2 岁，最后问第 1 个人多少岁，他说是 10 岁。请问第 5 个人多少岁？

6.2-0

```
#include<iostream>// 函数的递归（函数调用自身）
using namespace std;
int age(int n)
{
	if(n==1)
		return 10;
	else
		return	age(n-1)+2;// 调用自身
}
int main()
{
	cout<<age(5);
	return 0;
}
```

逆序字符

用 getchar 和 putchar 连续输入 5 个字符，逆序输出。

输入　　输入 5 个字符。　　输出　　逆序输出 5 个字符。

样例

输入　　abcde　　输出　　edcba

6.2-1// 逆序字符

```
#include <bits/stdc++.h>
using namespace std;

void reverse(int n)
{
	char next;
	if(n<=1)	// 递归的结束条件
	{
		next=getchar();// 从键盘获得一个字符
		putchar(next);// 因为只剩最后一个字符了，所以这里直接打印即可
	}
	else
	{
		next=getchar();// 从键盘获得一个字符
		reverse(n-1);  // 递归调用输入的下一个字符
		putchar(next);// 只有等到 n-1 个字符都被打印后，该字符才被打印
	}
}

int main()
{
	reverse(5);
	printf("\n");
	return 0;
}
```

逆序数

输入一个非负整数，输出这个数的逆序数（不考虑前导 0 的问题）。

输入　　输入一个 int 类型的非负整数。

输出　　输出这个数的逆序数。

样例

输入　　1000

输出　　0001

6.2-2// 逆序数

```
#include <bits/stdc++.h>
using namespace std;

int Turn(int n)
{
	if(n>=10)
	{
		printf("%d",n%10);
		Turn(n/10);
	}
	else
		printf("%d",n);
}

int main()
{
	int n;
	scanf("%d",&n);
	Turn(n);
	printf("\n");
	return 0;
}
```

求阶乘

用递归方法求 n!，例如当 n ＝ 5 时，n! ＝ 1×2×3×4×5 ＝ 120。

输入　　输入一个整数 n。

输出　　输出 n! 的值，保证结果不超过 64 位整数。

样例

输入　　5

输出　　5!=120

6.2-3//n 的阶乘

```
#include <bits/stdc++.h>
typedef unsigned long long ULL;//typedef 为 unsigned long long 创建别名为 ULL
using namespace std;
                                        //Factorial 翻译成中文是阶乘
ULL Factorial(int n)                    // 此处的 ULL 即为 unsigned long long
{
      if(n<0)
            exit(0);                            // 退出程序
      else if(n==0 || n==1)
            return 1;
            else
                  return Factorial(n-1)*n;
}

main()
{
      int n;
      scanf("%d",&n);
      printf("%d!=%llu\n",n,Factorial(n));// 注意 unsigned long long 的输出格式
      return 0;
}
```

最大公约数和最小公倍数

已知计算两个整数的最大公约数的递归公式是：

$$Gcd(m,n)=\begin{cases} m & n=0 \\ Gcd(n,m\%n) & n\neq 0 \end{cases}$$

两个整数的最小公倍数＝两个整数的乘积 / 两个整数的最大公约数，试求 n 个整数的最大公约数和最小公倍数。

输入

第一行输入一个整数 n（n\le12）n（n ≤ 12），表示有 n 个正整数（不超过 100）。 第二行输入 n 个整数。

输出

输出 n 个整数的最大公约数和最小公倍数，两数间以一个空格间隔。

样例

输入

4

9 12 30 15

输出

3 180

106012 - 复杂算式

已知$f（x，n）=\sqrt{n+\sqrt{(n-1)+\sqrt{(n-2)+\sqrt{...+2+\sqrt{1+x}}}}}$，输入$x$和$n$的值，计算$f（x，n）$的值。

输入

输入一个双精度浮点数x和一个整数$n（1 \leq n<100）$。

输出

f（x，n）的值，保留小数点后小数两位。

样例

输入	复制
3.6 10	
输出	复制
3.68	

6.2-5// 求平方根的递归解决

```
#include <bits/stdc++.h>
using namespace std;

double SquareRoot(double a,double x0)
{
	double x1=(x0+a/x0)/2;
	if(fabs(x1-x0)>1e-8)
		return SquareRoot(a,x1);
	else
		return x1;
}

int main()
{
	double x;
	scanf("%lf",&x);
	printf("%f\n",SquareRoot(x,1.0));// 尝试将 1.0 改为其他值试试？
	return 0;
}
```

某农场有一头刚出生的小奶牛，从第四个年头开始每年生一头母牛。之后的每一年大奶牛都会生一只小奶牛，而每过四年小奶牛就会长成大奶牛，长成大奶牛后又可以生小奶牛。请问 N 年后，此农场一共有多少头牛？

6.2-6// 母牛数

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ULL;//typedef 定义 long long 的别名为 ULL

ULL num[65];            // 此处的 long long 就可表示为 ULL

ULL Cow(int n)
{
	if(n<=4)
		return n;
	if(num[n])                    // 如果该值之前已经求出
		return num[n];              // 无需递归，直接取值
	else
		num[n]=Cow(n-1)+Cow(n-3);// 递归算出的值先存入 num[n]
		return num[n];              // 再返回结果 num[n]
}

int main()
{
	int n;
	while(scanf("%d",&n)==1)
	printf("%lld\n",Cow(n));
	return 0;
}
```

```
6.2-9// 汉诺塔
#include <bits/stdc++.h>
using namespace std;

void Move(int n,char A,char B,char C)// 将 n 个金片从 A 针借助 B 针移到 C 针
{
	if(n==1)                          // 当盘片只剩 1 个时
		printf("%c->%c",A,C);         // 移动金片从 A 到 C
	else
	{
		Move(n-1,A,C,B);                  // 递归调用，金片数 -1，三个针换位置
		printf("%c->%c",A,C);         // 移动金片从 A 到 C
		Move(n-1,B,A,C);                  // 递归调用，金片数 -1，三个针换位置
	}
}

int main()
{
	int m;                            // 金片数
	scanf("%d",&m);                   // 输入金片数
	Move(m,'A','B','C');
	return 0;
}
```