函 数

函数基础知识

6.0-1

```
#include<iostream>
using namespace std;
int jiafa(int a,int b)// 创建一个  加法  子函数
{
	return a+b;
}
int main()
{
	int x,y;
	cin>>x>>y;
	cout<<jiafa(x,y);// 主函数 main 调用子函数 jiafa
	return 0;
}
```

6.0-2

```
#include<iostream>
using namespace std;
int zuida(int a,int b)// 创建一个  选择大数  子函数
{
	if(a>b) return a;
	else return b;
}
int main()
{
	int x,y;
	cin>>x>>y;
	cout<<zuida(x,y);// 主函数 main 调用子函数 zuida
	return 0;
}
```

6.0-3

```
#include<iostream>
using namespace std;// 创建一个  乘法  子函数
void cheng(int a,int b);// 对子函数的声明，由于子函数在主函数 main 之后，所以需要事先声明
int main()
{
	int x,y;
	cin>>x>>y;
	cheng(x,y);
	return 0;
}
void cheng(int a,int b)//void 表示无返回值
{// 由于输出结果直接在子函数完成，无需返回结果值，因此用 void 表示无返回值
	cout<<a*b;
}
```

6.0-4

```
#include<iostream>
using namespace std;
int d;// 全局变量，可以在整个程序中使用
int jiafa(int a,int b)
{
	int c;//c 是局部变量，只能在子函数中使用
	c=a+b;
	return d-c; //d 是全局变量
}
int main()
{
	int x,y;//x,y 是局部变量，只能在主函数 main 中使用
	cin>>d>>x>>y;//d 是全局变量
	cout<<jiafa(x,y);
	return 0;
}
```

6.0-5

```
#include<iostream>
using namespace std;
int jiafa(int a,int b)
{
	int c;//c 是局部变量，只能在子函数中使用
	c=a+b;
	return c;
}
int main()
{
	int x,y;
	int c;//c 也是局部变量，与子函数的 c 名称相同，处于不同函数中，不产生冲突
	cin>>c>>x>>y;
	cout<<c-jiafa(x,y);
	return 0;
}
```

6.0-6-a

```
#include<iostream>// 数组函数，成功交换
using namespace std;
int i;// 全局变量，i 同时用于 mian 主函数和子函数，不产生冲突
void jiaohuan(int a[],int b[])
{
	int c;
	for(i=0;i<=2;i++)
	{	c=a[i];a[i]=b[i];b[i]=c;	}
}
int main()
{
	int e[]={1,2,3};
	int f[]={4,5,6};
	jiaohuan(e,f);//e,f 数组名
	for(i=0;i<=2;i++)		cout<<e[i]<<" ";
	return 0;
}
```

6.0-6-b

```
#include<iostream>// 数组函数错误举例，代码运行不成功，数据没有成功交换
using namespace std;
void jiaohuan(int a,int b){
	int c;
	c=a;a=b;b=c;
}
int main(){
	int e[]={1,2,3};
	int f[]={4,5,6};
	for(int i=0;i<=2;i++)	jiaohuan(e[i],f[i]);
	for(int i=0;i<=2;i++)		cout<<e[i]<<" ";
	return 0;
}
```

6.0-7

```
#include<iostream>// 二维数组函数
using namespace std;
int max(int a[][4])	// 列数不可为空
{
	int max=a[0][0];
	for(int i=0;i<3;i++)
	{
		for(int j=0;j<4;j++)
		{
			if(a[i][j]>max)	max=a[i][j];
		}
	}
	return max;
}
int main()
{
	int a[][4]={1,2,3,4,5,6,7,8,9,-1,-2,-3};
	cout<<max(a);
	return 0;
}
```

自定义函数练习题

选自东方博宜

1039. 求三个数的最大数

问题描述

已知有三个不等的数，将其中的最大数找出来。

输入：输入只有一行，包括 3 个整数。之间用一个空格分开。

输出：输出只有一行（这意味着末尾有一个回车符号），包括 1 个整数。

样例

输入复制：1 5 8

输出复制：8

```
#include<iostream>//2-1039
using namespace std;
int zuida(int a,int b,int c)// 创建一个  选择大数  子函数
{
	// 找出最大的数
	if(a > b && a > c)	return a;
	if(b > a && b > c)	return b;
	if(c > a && c > b)	return c;
}
int main()
{
	int x,y,z;
	cin>>x>>y>>z;
	cout<<zuida(x,y,z);
	return 0;
}
```

1258. 求一个三位数

求这样一个三位数，该三位数等于其每位数字的阶乘之和， 即 abc = a! + b! + c!。

(n! 表示 n 的阶乘，n!=1 * 2 * 3 * …… * n，如：5!=1 * 2 * 3 * 4 * 5)

```
#include<bits/stdc++.h>//3-1258
using namespace std;
int jiecheng(int x)
{
	int i,b,s,g,bb,ss,gg;
	b = x/100;		// 对当前 i 拆位
	s = x/10%10;
	g = x%10;
		// 分别计算 b、s、g 的阶乘
	bb = 1;
	ss = 1;
	gg = 1;
	for(i=1;i<=b;i++)	{	bb = bb * i;	}
	for(i=1;i<=s;i++)	{	ss = ss * i;	}
	for(i=1;i<=g;i++)	{	gg = gg * i;	}
	return bb + ss + gg;
}
int main()
{
	int i;
	for(i=100;i<=999;i++)
	{
		if( i == jiecheng(i) )
		{
			cout<<i<<endl;
		}
	}
	return 0;
}
```

1511. 数字之和为 13 的整数

求出 1 ~ n 范围内的整数，使其数字之和为 13，请问这样的数有多少个？

例如：数 85 ，其数字之和为 8+5=13 ；数 373，其数字之和为 3+7+3=13 。

输入：一个整数 n（n <= 10000000）；

输出：输出一个整数，代表符合条件数的总个数。

样例

输入 :1000

输出 :75

```
#include <bits/stdc++.h>//4-1511
using namespace std;
int he(int x)
{
	int s=0;
	while(x!=0)
	{
		s = s + x%10;
		x = x/10;
	}
	return s;
}
int main()
{
	int n,i,c=0;
	cin>>n;
	for(i=1; i<=n; i++)
	{
		if(he(i)==13)
			c++;
	}
	cout<<c;
	return 0;
}
```

1019. 求 1!+2!+...+N!

N!=1 * 2 * …… * N;

例如：5!=1 * 2 * 3 * 4 * 5=120 ；

编程求 1!+2!+3!+ …… +N!；

输入：输入一行，只有一个整数 n（ 1 ≤ n ≤ 10)；

输出：输出只有一行，包括 1 个整数。

输入：3

输出：9

```
#include <bits/stdc++.h>//5-1019  javacn
using namespace std;
int jc(int x)
{
	int r=1;
	int i;
	for(i=1;i<=x;i++)
		r=r*i;
	return r;
}
int main()
{
	int n;
	cin>>n;
	int i,sum=0;
	for(i=1;i<=n;i++)
	{
		sum=sum+jc(i);
	}
	cout<<sum;
	return 0;
}
```

1058. 求出 100 至 999 范围内的所有水仙花数。

所谓水仙花数，就是指各位数字立方之和等于该数的数；a^3 称为 a 的立方，即等于 a * a * a 的值。

例如：因为 153=1^3+5^3+3^3 ，所以 153 是一个水仙花数。

输入无

输出若干行，每行一个整数，表示该范围内的所有水仙花数。

按从小到大的顺序输出。

```
#include<bits/stdc++.h>//6-1058 求出 100 至 999 范围内的所有水仙花数
using namespace std;
int shui(int x)
{
	int  a = x / 100;
	int  b = x / 10 % 10;
	int  c = x % 10;
	return a*a*a+b*b*b+c*c*c;
}
int main()
{
	int i;
	for(i=100;i<=999;i++)
	{
		if(shui(i)==i)
		{
			cout<<i<<endl;
		}
	}
	return 0;
}
```

1445. 找亲戚

数字王国中，数字们也有亲戚关系。有一个 1 位数 x，他想找到自己的亲戚，他是这样判断对方是不是自己的亲戚的，如果对方的那个数的各个位中含有和自己一样的数，就认为对方是自己的亲戚。

比如： 3 和 635 就算亲戚，因为 635 中有数字 3 。

请从键盘读入一个一位的整数 x ，找出从 m ~ n 中有多少个数是 x 的亲戚。

```
#include<bits/stdc++.h>//7-1445
int x;// 全局变量
using namespace std;
int qinqi(int xx)
{
	int f=0;
	while(xx>0)
	{
		if(xx%10==x)
		{
			f=1;
			break;
		}
		xx=xx/10;
	}
	return f;
}
int main()
{
	int m,n,count=0;
	cin>>x>>m>>n;
	for(int i=m; i<=n; i++)
	{
		if(qinqi(i)) count++;
	}
	cout<<count;
	return 0;
}
```

输入：第一行，一个一位整数 x（x 是 1~ 9 之间的整数）

第二行，两个整数 m 和 n（m 和 n 也是 0 ~ 9999 之间的整数，且 m <= n）。

输出：一个整数，代表 x 的亲戚有多少个数。

输入：

1

1 10

输出 :2

```
#include<bits/stdc++.h>//1-1137  纯粹素数 13202828973
using namespace std;
bool sushu(int n)//判断素数函数
{
	bool f = true;
	for(int i = 2;i <= sqrt(n);i++)
	{
		if(n % i == 0)
		{
			f = false;
			break;
		}
	}
	//判断特殊情况
	if(n <= 1)		f = false;
	return f;
}

int main()
{
	int k=0;
	for(int i=1000;i<=3000;i++)
	{
		if(sushu(i)==true&&sushu(i%1000)==true&&
sushu(i%100)==true&&sushu(i%10)==true)
		{
			cout<<i<<" ";
		}
	}
	return 0;
}
```

```cpp
#include <bits/stdc++.h>//2-1139 孪生素数 javacn
using namespace std;
// 素数判断
bool sushu(int n)
{
	bool r = true;// 假设是素数
	for(int i = 2;i <= sqrt(n);i++)	// 找 n 因子范围
	{
		if(n % i == 0)
		{
			r = false;
			break;
		}
	}
	if(n <= 1) r = false;
	return r;
}

int main()
{
	int i,n;
	cin>>n;
	for(i = 2;i <= n-2;i++)
	{
		if(sushu(i) == true && sushu(i+2) == true )
		{
			cout<<i<<" "<<i+2<<endl;
		}
	}
	return 0;
}
```

定义函数，求一个数各个位的和（含 1，不含 n），循环 1~n 逐个判断是否是完全数。

```
#include <bits/stdc++.h>//3-1150  完全数 javacn
using namespace std;
int sum(int n){          /*求 n 的因子和（包括 1，不含 n）*/
      int r = 0;         //存放因子和
      for(int i = 1;i <= sqrt(n);i++)
      {
            if(n % i == 0)
            {
                  if(i == n / i)//如果因子相等
                  {
                        r = r + i;
                  }
                  else
                  {
                        r = r + i + n / i;
                  }
            }
      }
      r = r - n;//去掉 n
      return r;
}
int n,c = 0;
int main(){
      cin>>n;
      for(int i = 1;i <= n;i++)
      {
            if(sum(i) == i)
            {
                  c++;
            }
      }
      cout<<c;
      return 0;
}
```

```cpp
#include <bits/stdc++.h>//4-1513  绝对素数 13202828973
using namespace std;
bool sushu(int n)
{ //判断素数函数
	bool f = true;
	for(int i = 2;i <= sqrt(n);i++)
	{
		if(n % i == 0)
		{
			f = false;
			break;
		}
	}
	//判断特殊情况
	if(n <= 1) f = false;
	return f;
}

int main()
{

	for(int i=10;i<=99;i++)
	{
		if(sushu(i)==true&&sushu(i%10*10+i/10)==true)
		{
			cout<<i<<endl;
		}
	}
	return 0;
}
```

1514. 数根

数根是这样定义的：对于一个正整数 n ，将它的各个数位上的数字相加得到一个新数，如果这个数是一位数，我们就称之为 n 的数根，否则重复处理直到它成为一个一位数。

例如，n=34，3+4=7， 7 是一位数，所以 7 是 34 的数根。

再如，n=345，3+4+5=12，1+2=3，3 是一位数，所以 3 是 345 的数根。

对于输入数字 n ，编程计算它的数根。

输入：一个正整数。（n<=10^8）

输出：一个整数，代表 n 的数根。

输入：345

输出：3

```
#include<bits/stdc++.h>//5-1514    数根    javacn
using namespace std;
int qiuhe(int n) // 定义函数   求一个数的各个位上的和
{
	int s=0;//s 变量存放数的每位的和
	while(n!=0)		// 通过短除法求
	{
		s=s+n%10;
		n=n/10;
	}
	return s;
}
int main()
{
	int n;
	cin>>n;
	//n 变量如果是一位数，树根就是自己   否则要不停的求和直到是个位数
	while(n>=10)
	{
		n = qiuhe(n);
	}
	cout<<n<<endl;
	return 0;
}
```

```cpp
#include<bits/stdc++.h>//6-1512 甲乙的年龄 javacn
using namespace std;
bool sushu(int n){      // 定义函数：判断一个整数是否是素数
	bool r = true;     // 思路：假设是素数，循环找 n 的因子，找到因子就不是素数
	int i;
	for(i = 2;i <= sqrt(n);i++)
	{
		if(n % i == 0)          // 如果找到 n 的因子
		{
			r = false;
			break;
		}
	}
	if(n <= 1)     r =false;        // 素数是大于 1 的自然数如果没有因子
	return r;
}
int main(){
	int i,j,m,g,s;//i 代表乙的年龄   j 代表甲的年龄
    // 因为年龄和是两位数，且甲比乙大 13   所有乙的范围是 1-86
	for(i=1;i<=86;i++)
	{
		j = i + 13;
		m = i + j;
		if(m>=14&&m<=99&&sushu(m)==true)
		{
			g = m % 10;
			s = m / 10;
			if(g+s==13)
			{
				cout<<j<<" "<<i<<endl;
			}
		}
	}
    return 0;
}
```

```cpp
#include<bits/stdc++.h>//7-1151  桐桐数 javacn
using namespace std;
bool sushu(int n)
{
	bool r = true;
	int i;
	for(i = 2;i <= sqrt(n);i++)
	{
		if(n % i == 0)
		{
			r = false;
			break;
		}
	}
	if(n <= 1)	r =false;
	return r;
}
int main(){
	int n;
	cin>>n;
	bool f = false;// 假设不是桐桐数
	for(int i=2;i<=sqrt(n);i++)    // 循环找 n 的因子 从 2-sqrt(n)
	{
		if(n%i==0&&sushu(i)==true&&sushu(n/i)==true)
		{
			f = true;
			break;
		}
	}
	if(f == true)
		cout<<"It’ s a Tongtong number."<<endl;
	else
		cout<<"It’ s not a Tongtong number."<<endl;
	return 0;
}
```

```cpp
#include <bits/stdc++.h>//8-1143  纯粹合数  javacn
using namespace std;
// 定义函数：判断一个数是否是合数
bool heshu(int n)
{
	bool f = false;// 假设不是合数
	for(int i=2; i<=sqrt(n); i++)
	{
		if(n % i ==0)
		{
			f = true;
			break;
		}
	}
	//1 既不是素数也不是合数  我们假设不是合数
	// 所有 1 2 3 这些不进循环的都是满足的  不需要单独讨论
	return f;
}
int main()
{
	int i,j,k;
	for(i=100;i<1000;i++)
	{
		j = i / 10;
		k = i / 100;
		if(heshu(i) == true && heshu(j) == true && heshu(k) == true)
		{
			cout<<i<<endl;
		}
	}
	return 0;
}
```

思路一：遍历 1 到 n 的数，逐个调用函数判断回文

```
#include <bits/stdc++.h>//9-1149 回文数 javacn   推荐学习
using namespace std;

// 用于反读数字，返回倒序后的 x
int fun(int x)
{
	int r=0;
	while(x!=0)
	{
		r = x%10+r*10;
		x = x/10;
	}
	return r;
}

int main()
{
	int n,i,num=0;//num 计数回文数的个数
	cin>>n;
	for(i=1;i<=n;i++)
	{
		// 如果正读和反读一样就是回文数
		if(i==fun(i))
		{
			num++;
		}
	}
	cout<<num;
	return 0;
}
```

```cpp
#include<bits/stdc++.h>//10-1063 素数的个数 javacn
using namespace std;
int sushu(int n) //定义函数，判断素数
{
	bool f = true;	//用来标记 n 是否是素数，假设是素数
	int i;
	for(i=2; i<=sqrt(n); i++)
	{
		if(n%i==0)
		{
			f = false;
			break;
		}
	}
	if(n <= 1) f = false;
	return f;
}
int main()
{
	int m,n,s=0,i;//s: 素数的个数
	cin>>m>>n;
	for(i=m;i<=n;i++)		//m~n 范围内寻找
	{
		if(sushu(i))
		{
			s++;
		}
	}
	cout<<s;
	return 0;
}
```

```cpp
#include<bits/stdc++.h>//11-1064  素数 javacn
using namespace std;
int sushu(int n) //定义函数，判断素数
{
	bool f = true;	//用来标记 n 是否是素数，假设是素数
	int i;
	for(i=2; i<=sqrt(n); i++)
	{
		if(n%i==0)
		{
			f = false;
			break;
		}
	}
	if(n <= 1) f = false;
	return f;
}
int main()
{
	int n,c=0,i;// 素数的个数
	cin>>n;
	for(i=1;i<=n;i++)
	{
		if(sushu(i))// 如果 i 是一个质数
		{
			cout<<i<<" ";
			c++;
			if(c%5==0)  // 如果总个数是 5 的倍数，多输出一个换行符
			{
				cout<<endl;
			}
		}
	}
	return 0;
}
```

完数判断

一个数如果恰好等于它的因子之和，这个数就称 为“完数”。

例如，6 的因子为 1、2、3，而 6＝1＋2＋3 ，因此 6 就是“完数”。又如，28 的因子为 1、2、4、7、14 ，而 28＝1＋2＋4＋7＋14 ，因此 28 也是“完数”。

编写一个程序，判断用户输入的一个数是否为“完数”.

输入：输入只有一行，即一个整数。

输出：输出只有一行，如果该数为完数，输出 yes，否则输出 no。

样例

输入 6

输出 yes

思路 :1 求输入数 n 的因子和 ;2 如果 n 等于 n 的因子和，输出 yes.

```
#include<bits/stdc++.h>//12-1859 完数 javacn
using namespace std;
int n,sum=0;
int yinzi(int n)
{
	for(int i=1;i<n;i++)
	{
		if(n%i==0)
		{
			sum+=i;
		}
	}
}
int main()
{
	cin>>n;
	yinzi(n);   // 求 n 的因子和
	if(sum==n)  cout<<"yes";      // 如果 n 等于 n 的因子和，输出 yes
	else        cout<<"no";
	return 0;
}
```

```cpp
#include<bits/stdc++.h>//13-1862  友好数 javacn
using namespace std;
int yinzihe(int n)// 定义函数：求一个整数的因子和（不包括本事）
{
	int r = 1;// 因子 1 是都有的 直接加进来了
	for(int i=2;i<=sqrt(n);i++)    // 所有从 2 开始找
	{
		if(n%i==0&&i!=n/i)
		{
			r = r + i + n/i;
		}
		else if(n%i==0&&i==n/i)
		{
			r = r + i;
		}
	}
	return r;
}
int main()
{
	int n,m;
	cin>>n>>m;
	if(yinzihe(n) == m && yinzihe(m) == n)
	{
		cout<<"yes"<<endl;
	}
	else
	{
		cout<<"no"<<endl;
	}
	return 0;
}
```

1083. 回文数

回文数的定义为：如果把一个数的各个数位上的数字颠倒过来得到的新数与原数相等，则此数是回文数。例： 7,22,131,2112,31013…… 都是回文数。对任意给出的一个整数 n ，经过一系列的处理，最后都能成为回文数。处理的方法是，该数加上它的颠倒数，

例如：n=176 ；第一次处理后 176+671 ＝ 847 ；

第二次处理后 847+748 ＝ 1595 ；第三次处理后 1595+5951 ＝ 7546 ；

第四次处理后 7546+6457 ＝ 14003 ；第五次处理后 14003+30041 ＝ 44044 ；

此时成为回文数，共进行 5 次处理。

问题：给出 n 后，求出使该数按照以上规则进行一系列处理后成为回文数的最少操作次数。

输入：n 一个整数（1 ≤ n ≤ 1000000 ）。

输出：使 n 成为回文数的最少处理次数。

若开始给出的 n 是回文数，则输出 0 （即不需任何处理）。

输入：67 输出：2

```
#include<bits/stdc++.h>//14-1083  javacn
using namespace std;
int fun(int n)// 求 n 倒过来的数
{
	int r = 0;
	while(n != 0)	// 短除法求 n 倒过来的数
	{
		r = r * 10 + n % 10;
		n = n / 10;
	}
	return r;
}
int main() {
	int n,c = 0;
	cin>>n;
	while(n != fun(n)){	// 如果 n 不是回文
		c++;
		n = n + fun(n);
	}
	cout<<c;
	return 0;
}
```

思路：

1、判断回文的方式：如果数 n==n 倒过来的数,因此需要一个函数,求 n 倒过来的数（短除法）

2、使用 while 循环重复描述的过程，直到 n 为回文数

1562. 加数

给出一个正整数 n（1 ≤ n ≤ 100000）。在 n 的右边加入 n 的一半，然后在新数的右边再加入 n 的一半的一半，一直进行，直到不能再加为止。

例如　n=37　　37 的一半为 18（取整数）加到 n 的右边成为

3718　　18 的一半为 9，加到新数的右边成为

37189　　9 的一半为 4，加到新数的右边成为

371894　　4 的一半为 2，加到新数的右边成为

3718942　　2 的一半为 1，加到新数的右边成为

37189421　　1 的一半为 0，加数结束，最后得到的数是一个 8 位的数

输入：整数 n　　输出：加数结束后新数的长度。

输入：37　　输出：8

计算出当前数的位数 + 当前数不断除 2 的位数的和，直到计算到 0。

```
#include <bits/stdc++.h>//1562    javacn
using namespace std;
int n;
// 计算一个整数的位数
int fun(int x){
	int r = 0;
	while(x != 0){
		r++;
		x /= 10;
	}
	return r;
}
int main(){
	int ans = 0;
	cin>>n;
	while(n != 0){
		ans += fun(n);
		n /= 2;// 加一半的位数
	}
	cout<<ans;
	return 0;
}
```

1425. 查找含有 x 的数

请从一组数中找出含有 x 的数，统计出这样的数总共有多少个，总和是多少？

输入：输入包含三行：

第一行为 N，表示这组整数总共有多少个数（N ≤ 1000）；

第二行为 N 个整数（这些整数都是 1～9999 之间的数），整数之间以一个空格分开；

第三行包含一个一位的整数 x（x 是一个 1～9 之间的一位数）

输出：输出一行，有 2 个整数用空格隔开，第一个整数代表含有 x 的数的总个数，第二个整数代表含有 x 的数的总和。

输入

5

12 28 190 36 1255

2

输出

3 1295

```
#include <iostream>//1425
using namespace std;
int a[1010],n,x,s = 0,c = 0;
void chazhao(int aa,int b){
	bool d=false;
	int t=aa;
	while(t != 0)
	{
		if(t % 10 == b)
		{
			d=true;
			break;
		}
		t = t / 10;
	}
	if (d==true)
	{
		c++;
		s=s+aa;
	}
}
int main() {
	cin>>n;
	for(int i = 0; i < n; i++)		cin>>a[i];
	cin>>x;
	for(int i = 0; i < n; i++)  chazhao(a[i],x); // 逐个判断是否含有 x
	cout<<c<<" "<<s;
	return 0;
}
```

1397. 完美的偶数？

完美偶数指的是，如果一个数本身是偶数，且这个数是偶数位的数，且这个数的各个位也是偶数，那么这个数就可以称为完美偶数；比如： 28 就是完美偶数，而 246 就不是，因为 246 是一个 3 位数。

请你编程求出，从键盘读入的 n 个数中，哪些数是完美的偶数，输出他们。

输入

第一行是一个整数 n（ n ≤ 100 ）；

第二行是 n 个整数（这些整数都是 1~9999 范围内的整数）。

输出

按顺序输出这 n 个数中的完美偶数，每个数一行。

输入

5

26 4286 228 32 1280

输出

26

4286

输入

6

1 20 89 6686 468 2888

输出

20

6686

2888

```cpp
#include<bits/stdc++.h>//1397  完美偶数
using namespace std;
bool oushu1(int n)//判断位数是否偶数
{
	bool r=true;
	int c=0;
	while(n!=0) {	c++;  n=n/10;	}
	if(c%2!=0)		r=false;
	return r;
}
bool oushu2(int n)//判断每个位数是否偶数
{
	bool r=true;
	int x;
	while(n!=0)
	{
		x=n%10;
		if(x%2!=0)  {	r=false;	return 0;	}
		n=n/10;
	}
	return r;
}
int main(){
	int i,n,a[100];
	cin>>n;
	for(i=0;i<n;i++)
	{
		cin>>a[i];
		if(a[i]%2==0&&oushu1(a[i])==true&&oushu2(a[i])==true)
		{
			cout<<a[i]<<endl;
		}
	}
	return 0;
}
```