【2.4.1.3　Summing Digits】

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    char temp[1005];   //输入整数以字符数组 temp 存储
    while ( cin >> temp ) {
        int sum = 0, count;   // sum：各位数的和
        int len = strlen(temp);   // len：输入整数的位数
        if ( temp[0] == '0' && len == 1 ) break; //输入为 0 的情况
        for ( int i = 0 ; i < len ; ++ i )   //字符转相应的整数，逐位相加
            sum += temp[i] - '0';
        while ( sum > 9 ) {   //各位数的和不是个位数，继续求各位数的和
            count = 0;
            while ( sum ) {
                count += sum%10;
                sum /= 10;
            }
            sum = count;
        }
        cout << sum << endl; //输出结果
    }
    return 0;
}
```

【2.6.2　Quicksum】

```
#include <iostream>
using namespace std;
char s[300];   //输入的字符串（数据包）
int main()
{
    while(gets(s)&&s[0]!='#')   //每次循环输入当前测试用例，‘#’为结束符
     {
        int Quicksum=0;   //Quicksum 值初始化
        for(int i=0;i<strlen(s);i++) //计算 Quicksum 值
            if(s[i]>='A'&&s[i]<='Z')
                Quicksum +=(s[i]-'A'+1)*(i+1);
        printf("%d\n",Quicksum);   //输出 Quicksum 值
     }
    return 0;
}
```

【2.5.1 Pascal Library】

```
#include <iostream>
#include <cstring>
using namespace std;
int att[510][110];   //校友的出席筹款晚宴的情况用二维数组 att 表示
int main(void){
    int n, d;   //校友数 n，晚宴场数 d
    int i, j;
    int flag; //校友出席筹款晚宴的场数
    while(cin>>n>>d , n != 0 || d != 0){   //外层循环，每次处理一个测试用例
        memset(att, 0, sizeof(att));
        for (i = 0; i < d; i++){   //校友的出席筹款晚宴的情况
            for (j = 0; j < n; j++){
                cin>>att[i][j];
            }
        }
        for (j = 0; j < n; j++){   //对每个校友出席筹款晚宴的场数进行计算
            flag = 0;
            for (i = 0; i < d; i++){
                if (att[i][j] == 1){
                    flag++;
                }
            }
            if (flag == d){ //有校友参加了所有的晚宴
                break;
            }
        }
        if (flag == d){   //输出结果
            cout<<"yes"<<endl;
        }
        else{
            cout<<"no"<<endl;
        }
    }
    return 0;
}
```

【5.1.2 Train Swapping】

```
#include <iostream>
using namespace std;
int main() {
	int n;	//测试用例数目
	cin >> n;
	while(n--) {
		int m;	//车厢的数量
		int a[50];
		scanf("%d", &m);
		for(int i = 0; i < m; i++) {	//车厢的当前排列次序
			scanf("%d", &a[i]);
		}
		int x = 0;	//两个相邻车厢的最少的互换次数
		for(int i = 0; i < m - 1; i++)	//冒泡排序，累计互换次数
			for(int j = 0; j < m - i -1; j++)
				if(a[j] > a[j+1]) {
					int t = a[j];
					a[j] = a[j+1];
					a[j+1] = t;
					x++;
				}
		printf("Optimal train swapping takes %d swaps.\n", x);	//输出结果
	}
	return 0;
}
```

【3.1.1.1 Goldbach's Conjecture】

```
#include<cmath>
#include<cstring>
#include<cstdlib>
#include<cstdio>
using namespace std;
bool u[1111111];       //筛子
int su[1111111],num;     //素数表为 su[]，该表长度为 num
void prepare(){        //使用筛选法构建素数表 su[]
    int i,j,k;
    for(i=2;i<=1000000;i++)u[i]=true;
    for(i=2;i<=1000000;i++)     //顺序分析整数区间的每个数
    if(u[i]){                   //将 i 与当前素数的乘积从筛子中筛去
        for(j=2;j*i<=1000000;j++)
            u[j*i]=false;
    }
    for(i=2;i<=1000000;i++)if(u[i]){   //将筛中素数送入素数表
        su[++num]=i;
    }
}
int main () {
    prepare();      //使用筛选法构建素数表 su[]
    int i,j,k,n;
    while(scanf("%d",&n)>0&&n)     //反复输入偶整数，直至输入 0 为止
    {
        bool ok=false;
        for(i=2;i<=num;i++)   //按照递增顺序搜索素数表中的每个素数
        {
            if(su[i]*2>n)break;   //搜索完所有素数和的形式
            if(u[n-su[i]]){        //若 n 能够拆分出两个素数和的形式，则成功退出
                ok=true;
                break;
            }
        }
        if(!ok)puts("Goldbach's conjecture is wrong.");   //输出结果
        else printf("%d = %d + %d\n",n,su[i],n-su[i]);
    }
    return 0;
}
```