第六次练习赛题解

写在前面

不知不觉已经快要结课了，不知道大家程序设计掌握了多少。眼睛一闭一睁，期末考试要来了，希望大家都能取得一个好成绩。

下面给出几点建议：

1.认真把自己原本会的，却因为粗心大意*WA*的题总结原因。回顾一下犯的错误，比如输入没加*&*，*%c*输入了空格等空白字符，输出不加*\n*等问题。避免在考试时候犯类似的错误而又紧张找不到问题白白浪费时间。

2.你一定要学会独立找错，经过了一段时间的学习，相信你们应该具备了*debug*的能力，期末考试只能靠你自己。

3.熟知各种错误类型，方便你找错。（当然还是要祝你们一遍*AC*）

4.认真复习上课内容和题解。

5.程序设计是一门实践课程，做题才能保持手感！

6.别太紧张，心态很重要。

细心细心再细心，Ganten学姐用血教训告诉你们一定要细心。

A.Zyy学姐的生日礼物6(正版)

解题思路

本题较为简单，考察了冒泡排序和基础的字符串函数。*strcmp*本身就是按照字典序进行排序的，可以直接利用它进行比较。

AC代码

```
#include <stdio.h>
#include <string.h>
char s[1100][30],temp[30];

int main()
{
    int n,i,j,k;
    scanf("%d",&n);
    for(i=0;i<n;i++)
        scanf("%s",s[i]);
    for(i=0;i<n;i++)
    {
        for(j=i+1;j<n;j++)
        {
            if(strcmp(s[i],s[j])>0)
            {
                strcpy(temp,s[i]);
                strcpy(s[i],s[j]);
                strcpy(s[j],temp);
            }
        }
    }
    for(i=0;i<n;i++)
        printf("%s\n",s[i]);
    return 0;
}
```

B.为了虫群

解题思路

本题考查**二分答案**。如果对于某一个问题，其答案对于这个问题具有单调性，那么我们就可以考虑二分答案来求解。

这道题的实质是说对于x轴上的n个点，如果要用k个相同的圆去覆盖这n个点，圆的半径最小是多少。

我们假设现在换一种条件：如果用已知半径为R的若干个相同的圆去覆盖n个点，最少需要多少个圆。我们可以这样分析，要覆盖最左边的那个点，那么最左边的那个圆的圆心的最优放置位置一定距离最左边那个点为R，因为这样可以在覆盖了最左边的点的前提下，尽量覆盖更多的点。这是一种贪心的思维。放置完最左边的圆之后，排除了被这个圆覆盖的所有点，剩下的点又可以按照这种思路去处理。

可以发现，如果我们要覆盖n个点，每个圆的半径R越大，我们需要圆的数目一定是单调不增的，这就是之前所说的答案具有单调性。于是我们可以二分答案去求解最优半径。

AC代码

```
#include <stdio.h>
int n,k;
double x[100005];

int can(double r)
{
    int tot=1;
    double last=x[1];
    for(int i=2;i<=n;i++)
        if(x[i]-last>r*2)
            tot++,last=x[i];
    return tot<=k;
}

int main()
{
    scanf("%d%d",&n,&k);
    for(int i=1;i<=n;i++) scanf("%lf",&x[i]);
    double l=0,r=1e7,mid,eps=1e-8;
    while(r-l>eps)
    {
        mid=(l+r)/2;
        if(can(mid)) r=mid;
        else l=mid;
    }
    printf("%.6f",mid);

    return 0;
}
```

C.线代小测

解题思路

此题考察的知识点主要为伴随矩阵、逆矩阵、行列式的计算。可利用

$$AA^* = det(A)$$

的性质。这里使用按第一行展开计算的行列式。这种方法复杂度为指数级，但由于题目中矩阵较小可以完成计算。对于较大的矩阵需使用初等行变换法计算逆矩阵、行列式。

AC代码

```
#include<stdio.h>
int a[7][7], n;
double a1[7][7], a2[7][7];
//计算det时跳过的行、列
int flagr[7], flagc[7];
double det(int c){
    double ans = 0;
    int i =0, r = 0;//r：位于子矩阵第r行
    int cnxt = c+1;
    while(flagc[cnxt])cnxt++;
    for(i = 0; i < n; i++){
        if(flagr[i])continue;
        if(cnxt >= n) return a[c][i];
        flagr[i] = 1;//子矩阵跳过此行
        //
        ans += r & 1 ? -det(cnxt) * a[c][i]:det(cnxt)*a[c][i];//按行展开
        flagr[i] = 0;//下次计算前重置
        r++;
    }
    return ans;
}
void initF(){
    memset(flagr, 0, 8 * sizeof(flagr[0]));
    memset(flagc, 0, 8 * sizeof(flagc[0]));
}
int main(){
    scanf("%d", &n);
    int i, j;
    for(i = 0; i < n; i++)
        for(j = 0; j < n; j++)
            scanf("%d", &a[i][j]);
    //A*
    for(i = 0; i < n; i++){
        for(j = 0; j < n; j++){
            initF();
            flagr[i] = 1; flagc[j] = 1;
            //去掉第i行、第j列的子矩阵行列式
            //伴随矩阵不要忘记符号项
            a1[i][j] = ((i + j) & 1? -1:1) * det(j == 0 ? 1 : 0);
            printf("%.4lf ", a1[i][j]);

        }
        puts("");
```

```
    }
    //A-1
    puts("");
    initF();
    double detA = det(0);
    for(i = 0; i < n; i++){
        for(j = 0; j < n; j++){
            a2[i][j] =  a1[i][j] / detA;
            printf("%.4lf ", a2[i][j]);
        }
        puts("");
    }
}
```

D.蛇与数

解题思路

*n*的数量级是*7e7*，考虑一个一个数。观察一番规律可以发现，设第*i*个数为*n/m*，若*m*为*1*且*n+m*为偶数，下一个数为*(n+1)/m*，若*n*为*1*且*n+m*为奇数，下一个数为*n/(m+1)*，其他情况下，第*i+1*个数为*(n+(-1)^{n+m})/(m+(-1)^{n+m})*。

AC代码

```
#include<stdio.h>
#include<math.h>

int main() {
    int n,x = 1,y = 1,i;
    scanf("%d",&n);
    for(i = 2;i <= n; i++)
        if (y == 1 && (x+y)%2 == 1)
            x++;
        else if (x == 1 && (x+y)%2 == 0)
            y++;
        else if ((x+y)%2 == 0)
            x--,y++;
        else
            x++,y--;
    printf("%d/%d",x,y);
}
```

E.字符串比较

解题思路

一般情况下字符串比较定义为对其字典序的比较，本题是对库函数 `strcmp` 的考察。

另，建议当可以使用 `scanf` 时减少 `gets` 等的使用。

AC代码

```
#include<string.h>
#include<stdio.h>
char a[1000005],b[1000005];
int main()
{
    scanf("%s%s",a,b);
    int res=strcmp(a,b);
    if(res<0)puts("<");
    else if(res>0)puts(">");
    else puts("=");
    return 0;
}
```

F.HugeGun学姐的曲线

解题思路

本题更像是一道数学题，同学们在做的时候最重要的是要考虑到所有的边界情况：也就是什么情况下交点不存在。**向量法**相对而言实现起来较为简洁，就是因为两直线平行和重合的情况可以合并成一种，并且运算过程也不涉及除数为0的情况。这道题也启发我们对于其他的编程题，一个**清晰明确简单可实现**的思路要比上来就写好几十行代码**重要的多**，请同学们多加体会。

当然本题也有其他的实现方式，这是因为数学里面求两直线交点的解法本身就很多样，请同学们自行研究。并且建议比较一下不同方法的优劣，比如它们对于边界情况的处理等。

AC代码

```
/*
 Author: 冯旭杰(34728)
 Result: AC Submission_id: 2068859
 Created at: Sun Nov 17 2019 20:16:08 GMT+0800 (CST)
 Problem: 1513  Time: 6 Memory: 1536
*/
#include<stdio.h>
int main()
{
    int x1,y1,x2,y2,x3,y3,x4,y4,a1,b1,c1,a2,b2,c2,d,d1,d2;
    scanf("%d%d%d%d%d%d%d%d",&x1,&y1,&x2,&y2,&x3,&y3,&x4,&y4);
    a1=y1-y2;b1=x2-x1;c1=x1*y2-x2*y1;
    a2=y3-y4;b2=x4-x3;c2=x3*y4-x4*y3;
    if(a1*b2==a2*b1){puts("Zyysb");return 0;}
    d=a1*b2-a2*b1;d1=c2*b1-c1*b2;d2=a2*c1-a1*c2;
    printf("(%.6lf,%.6lf)",(double)d1/d,(double)d2/d);
}
```

G.kth derivative

解题思路

本题读入字符串后可以对每一项分别进行求导处理，分为以下几种情况：1、求导之后仍有*x*存在；2、求导之后为一个常数；3、求导之后没有此项。针对每种情况分别进行处理即可，其中容易出问题的特殊情况是对*x*求一阶导数，因为系数为1且次数项恰好为0，在最终输出环节可能被忽略。

AC代码

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
char s[110];
int a,coe,i=0,j,k,expo,flag,flg,fl=0,f=0;//a数字,coe参数,expo指数,四个标志
int main()
{
    scanf("%d%s",&k,s);
    while(i<strlen(s))
    {
        flag=0;a=0;flg=0;
        if(s[i]=='-')   flg=1,i++;
        if(s[i]=='+')   i++;
        while(isdigit(s[i]))
        {
            a*=10;
            a+=s[i]-'0';
            flag=1;
            i++;
        }
        if(s[i]=='x')
        {
            if(flag==1&&flg==1)
                coe=(-1)*a;
            else if(flag==1)
                coe=a;
            else if(flg==1)
                coe=-1;
            else coe=1;
            if(i<strlen(s)) i++;
        }
        else    continue;
        flag=0;a=0;
        if(s[i]=='^')
        {
            i++;
            if(s[i]=='-')   flag=1,i++;
            while(isdigit(s[i]))
            {
                a*=10;
                a+=s[i]-'0';
                if(i<strlen(s)) i++;
            }
            if(flag==0) expo=a;
            else expo=(-1)*a;
```

```
            a=0;
        }
        else expo=1;
        if(expo>0&&expo<k)  continue;
        else
        {
            for(j=0;j<k;j++)
            {
                coe*=(expo-j);
            }
            expo-=k;
        }
        if(coe>0&&fl==1) printf("+"),f=1;
        if(coe!=1||expo==0) printf("%d",coe),fl=1,f=1;
        if(expo==0) continue;
        else if(expo!=1)    printf("x^%d",expo),f=1;
        else printf("x"),f=1;
        if(s[i]=='-')   continue;
    }
    if(f==0) printf("0");
    return 0;
}
```