第三次上机题解

前言

请大家不要太在意一次上机的成绩，这次的题平均起来比较难，但题目质量是可以保证的，认真学习老师上课内容的同学基本可以做出来2-3题，额外认真学习课件的同学能完成4-5题，练习赛有认真练习的同学能再多出至少2道，课下花了更多功夫的同学能做出来更多。

一定要学会总结错误经验，很多公告、hint和题解其实都是我们学习这门课的血泪教训，请务必仔细研读题解思路和代码。

建议同学们学习有关调试的技巧，自己多测试几组数据再交，这对提升做题效率有很大帮助。 课下学习的时候，请学会使用搜索引擎搜索各种问题，而不是简简单单地给同学贴一个代码，“请帮我看看这是什么问题”。

课上做题，请务必仔细读题（内容及格式都要仔细阅读），对没有认真阅读空格换行等格式问题导致的错误以后将不再强调。

大家如果能抽出课余时间，在理解课件内容的基础之上多加动手实践、练习，一定能取得更加满意的成果！ P.S.有什么建议和意见，欢迎私戳各个助教提出！

A.十进制转三进制

解题思路

本题考察了多组数据读入和基本的进制转换。 对于不定组数的多组输入，可以使用类似`while (~scanf("%d", &n))`，或`while (scanf("%d", &n))`，甚至`while (scanf("%d", &n) + 1)`的语句来实现。

对于进制转换，我们可以使用数组+循环模拟不断做短除法的过程，然后倒序输出余数即可。

AC代码

```
#include <stdio.h>
#define MN 1024

int stack[MN];
int main()
{
    int top, n;
    while (~scanf("%d", &n))
    {
        top = 0;
        do stack[top++] = n % 3;
        while (n /= 3);

        while (top--)
            printf("%d", stack[top]);
        puts("");
    }

    return 0;
}
```

B. Potassium种菜

解题思路

考查数组应用。

使用一个数组a存储菜的高度（$a[i]$表示第i棵菜的高度），依据题意对高度进行修改和输出即可。

注意数组大小：题目中说过了$1 \leq n \leq 100000$，故开100010左右即可。

AC代码

```
#include<stdio.h>
int a[100010];
int main(){
    int i,n,q,opt,x,h,k;
    scanf("%d%d",&n,&q);
    for(i=1;i<=n;i++)scanf("%d",&a[i]);
    while(q--){
        scanf("%d",&opt);
        if(opt==1){
            scanf("%d%d",&x,&h);
            a[x]+=h;
        }else{
            scanf("%d",&k);
            while(k--)scanf("%d",&x),printf("%d ",a[x]);
            puts("");
        }
    }
    return 0;
}
```

C. 301 Moved Permanently

解题思路

仔细阅读PPT有关文件输出重定向知识即可做出本题。

一个小$trick$:均使用浮点数读入，视情况输出。

AC代码

```
#include<stdio.h>
int main(){
    freopen("301.txt","w",stdout);
    int i,n,opt;
    double a,b;
    scanf("%d",&n);
    while(n--){
        scanf("%d%lf%lf",&opt,&a,&b);
        if(opt==3)printf("%.4f",a+b);
        else printf("%.0f",a+b);
        puts("");
    }
    fclose(stdout);
    return 0;
}
```

D 再解一元二次方程

解题思路：

这道题考察的主要是浮点数精度的问题，在运行过程中要两次将所需数值与0进行比较，考虑到double类型的数字在运行过程中存在精度上的问题，我们需要引入eps来辅助进行判断。如果直接使用形如a == b这样的精确判断，可能会使得原本相等的两个数被判断为不相等，这是我们所不希望看到的，因而我们使用形如fabs(a) < eps这样的判断方式进行判断。这个代码运行过程中的两次比较实际上是有所区别的，因为直接输入的0是精确数值，不必须使用eps来进行辅助判断。

考察到的格式化输出是常见类型，如何输出需要同学们进行一定的记忆。

AC代码：

```
#include <stdio.h>
#include <math.h>
#define eps 1e-8
int main()
{
    double a, b, c, d;
    double x1, x2;
    scanf("%lf%lf%lf", &a, &b, &c);

    if(fabs（a) < eps) printf("Not a quadratic");
    //这里也可以写成if(a == 0) printf("Not a quadratic");因为直接读入一个0应当是精确的
    else
    {
        d = b * b - 4 * a * c;
        if(d > eps)
        {
            d = sqrt(d);
            x1 = (-b + d) / (2 * a);
            x2 = (-b - d) / (2 * a);
            if(a > 0) printf("Two unequal real roots:%.6f,%.6f", x2, x1);
            else printf("Two unequal real roots:%.6f,%.6f", x1, x2);
        }
        else if(d < -eps)
        {
            d = sqrt(-d);
            x1 = -b / (2 * a);
            x2 = d / (2 * a);
            printf("Two conjugate complex roots:%.5f+%.5fi %.5f-%.5fi", x1, x2,
x1, x2 );
         }
        else printf("Two equal roots:%10.5f", -b / (2 * a));
    }
}
```

另解：由于在判断两根谁大的时候，分母a会产生影响，故先钦定a为正的（如果为负的那么整个方程乘上一个负号），后面就无需讨论了。

```
#include<stdio.h>
#include<math.h>
int main(){
```

```
    double a,b,c;
    scanf("%lf%lf%lf",&a,&b,&c);
    if(a<0)a=-a,b=-b,c=-c;
    if(fabs(a)<1e-10)printf("Not a quadratic");
    else{
        double delta=b*b-4*a*c;
        if(delta<-1e-7){
            double x=-b/(2*a),y=sqrt(-delta)/(2*a);
            printf("Two conjugate complex roots:%.5f+%.5fi %.5f-%.5fi",x,y,x,y);
        }else if(fabs(delta)<1e-7){
            printf("Two equal roots:%10.5f",-b/(2*a));
        }else{
            double x1=(-b-sqrt(delta))/(2*a),x2=(-b+sqrt(delta))/(2*a);
            printf("Two unequal real roots:%.6f,%.6f",x1,x2);
        }
    }
    return 0;
}
```

E. 士谔1973

解题思路

先预处理，标记出所有可以输出的数（最开始，所有士谔数都是未输出的士谔数）。

每次询问一个区间的时候，枚举区间内的每个数，如果某个数可以输出，那么将它输出，并把这个数标记为不可以输出的数。

AC代码

```
#include<stdio.h>
#define N 100000
int can[N+5];
int main(){
    int i,n,l,r;
    for(i=0;i<=N;i++)if(i%10==1||i%10==9||i%10==7||i%10==3)can[i]=1;
    scanf("%d",&n);
    while(n--){
        scanf("%d%d",&l,&r);
        for(i=l;i<=r;i++)
            if(can[i])
                printf("%d ",i),can[i]=0;
        puts("");
    }
    return 0;
}
```

F.二进制翻转

解题思路

本题考察了循环和位运算。

对于32位整数，我们可以进行32次循环，从低位到高位逐位获取其二进制各位上的数，然后再从高位到低位地累计到一个初始化为0的变量上，就得到位翻转的结果了。

AC代码

```
#include <stdio.h>

int main()
{
    int n, i, rev = 0;
    scanf("%d", &n);

    for (i=0; i<32; ++i)
    {
        rev <<= 1;
        rev = rev | (n & 1);
        n >>= 1;
    }

    printf("%d", rev);
    return 0;
}
```

另解：这个数的二进制形式可以看做一个$x = 2$的多项式，通过秦九韶算法直接得到答案：

```
#include<stdio.h>
int main(){
    int i,n,m=0;
    scanf("%d",&n);
    for(i=0;i<32;i++)m=(m*2)+(n%2),n/=2;
    printf("%d",m);
    return 0;
}
```

G.光线反射

解题思路：

首先，我们要考虑是否会发生反射，发生反射时改变的那一数据，这时我们得到反射光的方向向量，再将其单位化就可以得到我们需要的答案。

此题要求不能输出-0.00，考虑到题目要求的输出范围，我们所设置的eps不能过小，根据要求要保留两位小数，而c语言在%.2f这样的格式化输出中会直接进行四舍五入，所以我们应该将eps设为5e-3。

AC代码：

```
#include <stdio.h>
#include <math.h>
#define eps 5e-3
int main()
{
    int n, d;
    double x, y, z, dis;
    char a;
    FILE *fp;
//  fp = fopen("out.txt","w");

    scanf("%d", &n);
    while(n--)
    {
        scanf("%lf%lf%lf %c%d",&x ,&y ,&z, &a, &d);
        //%c前必须填写空格否则a将读取到输入的空格而非我们所要的字符
        switch(a)
        {
            case 'x':
                if(x * d > 0) x = -x;
                break;
            case 'y':
                if(y * d > 0) y = -y;
                break;
            case 'z':
                if(z * d > 0) z = -z;
        }
        dis = sqrt(x * x + y * y + z * z);
        x = fabs(x / dis) < eps ? 0 : x / dis;
        y = fabs(y / dis) < eps ? 0 : y / dis;
        z = fabs(z / dis) < eps ? 0 : z / dis;
        printf("%.2f %.2f %.2f\n", x, y, z);
    }
 }
```

H.最小余数

解题思路

容易发现，如果满足 $R-L+1>=2018$，即区间长度不短于 2018 ，则区间内必有2019的倍数，此时答案必为0；否则，区间长度必小于2018，此时可直接写一个循环枚举求解而不用担心超时。

AC代码

```
#include <stdio.h>
#define YEAR 2018

int main()
{
    int L, R, ans, i, j, re;
    scanf("%d %d", &L, &R);

    if (R-L+1 >= YEAR)
        ans = 0;
    else
    {
        ans = YEAR;
        for (i=L; i<=R; ++i)
        {
            for (j=i+1; j<=R; ++j)
            {
                re = (i % YEAR) * (j % YEAR) % YEAR;
                if (ans > re)
                    ans = re;
            }
        }
    }

    printf("%d", ans);

    return 0;
}
```

I.Zyy学姐的生日礼物

解题思路

本题仍然考察了闰年的判断。

对于本题，数据其实保证了年份为正整数，可以分别把两个时间点都转换成“天数”，然后直接用两个时间点对应的天数相减，取绝对值即可。

AC代码

```
#include <stdio.h>

int MON_DAY[2][13] =
{
    {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31},
    {0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}
};
int YEAR_DAY[2]=
{
    365,
    366
};

int is_leap(int x)
{
    return (x%4==0 && x%100!=0) || (x%400 == 0);
}

int get_day(int y, int m, int d)
{
    int i, days = d;
    for (i=0; i<y; ++i)
        days += YEAR_DAY[is_leap(i)];
    for (i=1; i<m; ++i)
        days += MON_DAY[is_leap(y)][i];
    return days;
}

int main()
{
    int y1, m1, d1, y2, m2, d2, dist;
    scanf("%d %d %d %d %d %d", &y1, &m1, &d1, &y2, &m2, &d2);
    dist = get_day(y1, m1, d1) - get_day(y2, m2, d2);

    printf("%d\n", abs(dist));
    return 0;
}
```

而对于整数范围内的年份也是可解的：

假设前一个日期距离其对应年份（y）1月1日有$x1$天，后一个日期距离其对应年份（yy）1月1日有$x2$天，yy年的1月1日距离y年1月1日有dy天，那么这两个日期的差即为$abs(x2+dy-x1)$。

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

```c
#include<stdlib.h>
int day[]={0,31,28,31,30,31,30,31,31,30,31,30,31};
int run(int y){return y%400==0||(y%4==0&&y%100);}
int max(int a,int b){return a>b?a:b;}
int min(int a,int b){return a>b?b:a;}
int main(){
    int i,y,m,d,yy,mm,dd;
    scanf("%d%d%d",&y,&m,&d);
    scanf("%d%d%d",&yy,&mm,&dd);
    int x1=d,x2=dd,dy=0;
    day[2]=28+run(y);
    for(i=1;i<m;i++)x1+=day[i];
    day[2]=28+run(yy);
    for(i=1;i<mm;i++)x2+=day[i];
    for(i=min(y,yy);i<max(y,yy);i++)dy+=run(i)+365;
    if(y>yy)dy=-dy;
    printf("%d",abs(dy+x2-x1));
    return 0;
}
```

J.滚雪球

解题思路：

此题主要考察的是对字符串的一些处理，主要需要使用的是字符串连接strcat、字符串拷贝strcpy、字符串比较strcmp三个函数，将上一个人的字符连接在自己的字符之后进行比较就可以判断自己的输出是否正确，然后依次将当前人的输出放进last中保存。要注意字符串的长度，包括结尾的\0等，要开的足够长，否则会出现一些问题。

AC代码：

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

int main()
{
    int n;
    char last[4050], current[4050], ans[4050];
    //这里尤其需要注意的是current这个字符串，由于上一个同学的字符可以有4000，自己的字符串可以有20，加上\0拼接起来长度可以有4021
    //如果数组开的太小会超出数组范围会OE

    last[0] = '\0';//必须要初始化last，否则在拼接的时候会导致错误
    current[0] = '\0';//这个可以不初始化

    scanf("%d", &n);
    while(n--)
    {
        scanf("%s", current);
        strcpy(ans, current);
        strcat(ans, last);
        scanf("%s", last);
        if(strcmp(last, ans) == 0) printf("1\n");
        else printf("0\n");
    }
}
```

K. 字母频率统计

解题思路

记录下第x个字母出现的次数$cnt[x]$（如：'a'/'A'出现的次数为$cnt[0]$，'z'/'Z'出现的次数为$cnt[25]$），找出柱状图共需要输出多少行（$mx = \max_{x\in[0,25]} cnt[x]$）。

从上到下（$mx \to 1$）枚举当前高度h，当$h \geq cnt[x]$时，x对应列输出$'*'$，否则输出空格。

AC代码

```
#include<stdio.h>
#define max(a,b) ((a)>(b)?(a):(b))
int cnt[26],mx;
char c;
int main(){
    int i,j;
    while(~scanf("%c",&c)){
        if(c>='a'&&c<='z')cnt[c-'a']++;
        else if(c>='A'&&c<='Z')cnt[c-'A']++;
    }
    for(i=0;i<26;i++)mx=max(mx,cnt[i]);
    for(i=mx;i>=1;i--){
        for(j=0;j<26;j++){
            if(cnt[j]>=i)printf("*");
            else printf(" ");
        }
        puts("");
    }
    for(i=0;i<26;i++)printf("%c",'a'+i);
    return 0;
}
```

易错点(反面教材):

大家保存代码文件时后缀请选择`.c`而不是`.cpp`

```
scanf("lld",&a);
```

```
printf("/n");
```

```
int n;
int a[n];
```

```
i=0;
while(n)
{
    a[n++]=n%3;
    n/=3;
}
for(;i>0;i++)printf("%d",a[i]);
```

```
int a[10],i;
for(i=1;i<=10;i++)scanf("%d",&a[i])
```

```
scanf("%d %d",&a,&b);//这样写并不好
```

```
freopen(...,stdout);
printf(...);
fclose(stdout);
printf(...);
fclose(stdout);
printf(...);
fclose(stdout);
```

```
x1=-b+sqrt(a)/(2*a);
x2=(-b-sqrt(a))/2*a;
```

```
int a,b,c;
scanf("%d%d%d",&a,&b,&c);
double c=a/b*(double)c;
```

```
int a=pow(2,10);
```