第五次上机题解

前言：

本次题目考察的一些基本的语法和字符处理方式请同学们熟练掌握，受用一生。

思维的训练也是必须的，否则学习C语言的意义不大，别人几行能搞定的事情如果你要写上百行才能解决，你的竞争力何在？本次上机每道题代码量都不是很大，尽量在赛后补题，写得多会得多。

另外请不要背代码，这大概等于学英语只背字母，毫无作用。

A.成绩换算

解题思路：

使用`math.h`里的函数`sqrt`来对分数开方。

注意，`sqrt`计算的结果为`double`类型，在转为`int`时需要先乘10再转，否则`10*(int)(3.4)`答案为30，而正确答案`(int)(10*3.4)`答案为34，就会错，

AC代码：

```
#include<stdio.h>
#include<math.h>
int n,x;
int main()
{
    scanf("%d",&n);
    while(n--)
    {
        scanf("%d",&x);
        printf("%d\n",(int)(10*sqrt(x)));
    }
    return 0;
}
```

B.吃鱼密码

解题思路：

题目说这个字符串只有一行，这里可以采用循环读取单个字符的方式，判断字符串终止(EOF==-1)即可，也可以用`gets`等函数。

注意字符串中可能存在空格、制表符(\t)等空白字符，会导致`scanf(%s)`只读取前一部分。

转换字符的时候可以自己判断也可以使用以下库函数判断

```
isalpha();
isdigit();
isupper();
islower();
```

AC代码：

```
#include<stdio.h>
#include<math.h>
int main()
{
    char ch=getchar();
    while(ch!=-1)
    {
        if(ch>='0'&&ch<='9')putchar('0'+(9-(ch-'0')));
        else if(ch>='a'&&ch<='z')putchar(ch-'a'+'A');
        else if(ch>='A'&&ch<='Z')putchar(ch-'A'+'a');
        else putchar('_');
        ch=getchar();
    }
    return 0;
}
```

C. Ackman函数

解题思路：

递归实现，写一个函数`A`，判断属于三种情况中的哪一种返回相应的值即可。

AC代码：

```
#include<stdio.h>
int A(int m,int n)
{
    if(!m)return n+1;
    if(n==0)return A(m-1,1);
    return A(m-1,A(m,n-1));
}
int main()
{
    int T;
    scanf("%d",&T);
    while(T--)
    {
            int m,n;
            scanf("%d%d",&m,&n);
            printf("%d\n",A(m,n));
    }
    return 0;
}
```

D. Rock, Paper, Scissors, Lizard, Spock

解题思路：

简单陈述一下题面，石头剪刀布蜥蜴人斯波克五者之间相生相克，制约关系如题图。玩家a、b以周期性规律出招，循环节分别为NA、NB，获胜得一分，平局或输不得分，问玩n个回合两位玩家分别得分多少。

可以用一个二维数组存获胜关系，`rules[i][j]`为1表示i胜j,为0表示i不胜j。从1循环到n模拟每回合游戏，用两个变量j,k分别表示两个玩家当前出的招式，每回合判断输赢计数，之后j,k每次`++`，大于循环节时置为1即可。

AC代码：

```
#include<stdio.h>
int rules[5][5]=
{
    {0,0,1,1,0},
    {1,0,0,1,0},
    {0,1,0,0,1},
    {0,0,1,0,1},
    {1,1,0,0,0}
};
int a[505],b[505];
int main()
{
    int n,na,nb;
    scanf("%d%d%d",&n,&na,&nb);
    for(int i=1;i<=na;i++)scanf("%d",&a[i]);
    for(int i=1;i<=nb;i++)scanf("%d",&b[i]);
    int j=1,k=1,res1=0,res2=0;
    for(int i=1;i<=n;i++)
    {
        if(rules[a[j]][b[k]])res1++;
        if(rules[b[k]][a[j]])res2++;
        j++,k++;
        if(j>na)j=1;
        if(k>nb)k=1;
    }
    printf("%d %d\n",res1,res2);
    return 0;
}
```

E.最后一条毛毛虫

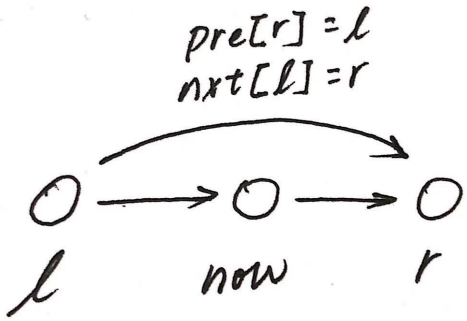
解题思路：

用链表模拟报数的行为即可。

设置$nxt[i]$表示编号为i毛毛虫的下一条毛毛虫编号，$pre[i]$表示上一条，用变量now表示当前报数的毛毛虫，向后数一个时`now=nxt[now]`。

循环n次，每次删去报到m的毛毛虫，第n次报到的即答案。

删除节点now的操作



AC代码：

```
#include<stdio.h>
int n,m,pre[3003],nxt[3003];
int main()
{
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++)pre[i]=i-1,nxt[i]=i+1;
    nxt[n]=1,pre[1]=n;
    int now=n;  //为什么设置now为n:nxt[n]=1,这样使得1号位第一次报1
    for(int i=1;i<=n;i++)
    {
        for(int j=1;j<=m;j++)now=nxt[now];  //向下数m次
        if(i==n)printf("%d\n",now);
        else
        {
            int l=pre[now],r=nxt[now];
            pre[r]=l,nxt[l]=r;
        }
    }
    return 0;
}
```

F. Ange的数组

解题思路：

本题考查点为数组。

设$S=\{1,2,\dots,n\}$，问题即为：找到最小的k，使得S的任意大小为k的子集T中存在两个不同的整数x,y，x是y的倍数。

容易发现极端情况为$T=\{n-k+1,\dots,n\}$，设$a,b\in T$，且$a<b$，若$2b\in T$，则$2a\in T$，所以T的最小元的2倍一定在T中。故令$2(n-k+1)\le n$，解得$2k\ge n+2$，即$k_{min}=\lfloor\frac{n+3}{2}\rfloor$。

若$k<k_{min}$，则$T=\{n-k+1,\dots,n\}$时，$2(n-k+1)>n$，T的最小元的2倍不在T中，故T的最小元的任意大于1的倍数都不在T中。同理，T中其他元素的任意大于1的倍数都不在T中。所以T中不存在两个不同的整数x,y，x是y的倍数。故$k\ge k_{min}$。

下证$k=k_{min}$时，S的任意大小为k的子集T中存在两个不同的整数x,y，x是y的倍数。

任何一个正整数都可以表示成一个奇数与2的次幂的积，即若$m\in N^*,k\in N^*,n\in N$，则$m=(2k-1)\cdot 2^n$，并且这种表示方式是唯一的。

1°当n为偶数时，$S=\{1,2,\dots,n\}$中共有$\frac{n}{2}$个奇数。

我们将S中的元素分成如下$\frac{n}{2}$个抽屉：

$T_1=\{1,1\times 2,1\times 2^2,\dots\}$

$T_2=\{3,3\times 2,3\times 2^2,\dots\}$

……

$T_{\frac{n}{2}}=\{\frac{n}{2}-1\}$

这样，S中的元素就无重复、无遗漏地放进这$\frac{n}{2}$个抽屉中了。从这n个数中任取k_{min}个数，即$\frac{n+2}{2}$个数，根据抽屉原理，必定至少有两个数属于同一个抽屉，显然，任何同一个抽屉中的两个数之间具有倍数关系。

2°当n为奇数时，S中共有$\frac{n+1}{2}$个奇数。同理构造$\frac{n+1}{2}$个抽屉可证。

综上，当$k_{min}=\lfloor\frac{n+3}{2}\rfloor$时，$S$的任意大小为$k$的子集$T$中存在两个不同的整数$x,y$，$x$是$y$的倍数。

（特别鸣谢：yuki舒）

AC代码：

```
#include <stdio.h>
int main() {
    int n;
    while (~scanf("%d", &n))
        printf("%d\n", (n + 3) / 2);
    return 0;
}
```

G.计算面积

解题思路：

用题面说的投针法，注意判断$sin(x^2)$和y的正负关系，还要注意最后面积的计算是要和$2n$比较。

另外，本题使用暴力积分和辛普森积分都可以求，有兴趣可以学习(自适应)辛普森积分，使用面很广，精度可调，速度很快(其实就是分段二次函数拟合)

AC代码：

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
double n;
int main()
{
    scanf("%lf",&n);
    int i,ans=0;
    for(i=1;i<=30000000;i++)
    {
        double x=1.*rand()/RAND_MAX*n;
        double y=1.*rand()/RAND_MAX*2-1;
        double temp=sin(x*x);
        if(((temp>=0)^(y>=0))==0&&fabs(y)<=fabs(temp))ans++;
    }
    printf("%.2f",1.*ans/30000000*n*2);
    return 0;
}
```

自适应辛普森积分：（有兴趣的同学可以搜索相关资料）

```
#include <stdio.h>
#include <math.h>
double eps=1e-5;
double f(double x) {return fabs(sin(x*x));}
double simpson(double l,double r) {return (r-l)*(f(l)+f(r)+f((l+r)/2)*4)/6;}
double solve(double l,double r,double ans)
{
    double mid=(l+r)/2;
    double a=simpson(l,mid),b=simpson(mid,r);
    if(fabs(a+b-ans)<eps) return a+b+(a+b-ans);
    return solve(l,mid,a)+solve(mid,r,b);
}
int main()
{
    double n;
    scanf("%lf",&n);
    printf("%.2f",solve(0,n,simpson(0,n)));
    return 0;
}
```

H.带数字比较

解题思路：

可以先处理一正一负的情况，负数永远小于整数。

剩下的情况我们先比较两个数绝对值的大小关系。可以将两个数倒过来存(将两个数的高位对齐)，然后从高位往低位比较，当然也可以先判断两个数的长度关系，长的比短的大。由于不知道是否有前导零，这里采用对齐高位的做法。

最后根据正负关系和绝对值大小关系得到两个数的大小关系。

AC代码：

```
#include<stdio.h>
#include<string.h>
char s[111],t[111];
int a[111],b[111],ans,n;
int main()
{
    scanf("%d",&n);
    while(n--)
    {
        scanf("%s%s",s,t);
        int lens=strlen(s),lent=strlen(t),fs=1,ft=1;
        if(s[0]=='-')fs=-1;
        if(t[0]=='-')ft=-1;
        if(fs<ft)puts("zd");
        else if(ft>fs)puts("xx");
        else
        {
            ans=0;/*1:A>B 2:A==B 3:A<B*/
            int i,l=0,r=(lens>lent?lens:lent)-1;/*l,r为需要比较的区间下标 */
            for(i=0;i<=r;i++)
            {
                if(i<lens)a[r-i]=s[i]-'0';
                else a[i]=0;/*清空不存在的更高位 */
                if(i<lent)b[r-i]=t[i]-'0';
                else b[i]=0;/*同上 */
            }
            r-=(fs==-1);/*负号在最高位 */
            for(i=r;i>=l;i--)
                if(a[i]>b[i]){ans=1;break;}
                else if(a[i]<b[i]){ans=-1;break;}
            if(fs==-1)ans=-ans;
            if(ans==1)puts("xx");
            else puts("zd");
        }
    }
    return 0;
}
```

I.倒立汉诺塔

解题思路：

汉诺塔倒立与否其实~~无关紧要~~原理相同。要将s上的n个盘子移动到t上依旧是先把上面的$(n-1)$个盘子移到辅助杆上，将s上的最下面一个盘子移到t上，再将辅助杆上的$(n-1)$个盘子移到t上。

题解代码中状态设计是low代表最小盘子的大小，$high$代表最大盘子的大小，s、t分别代表要移动盘子的起点和终点。递归处理，当`low==high`时需要移动的只有一个盘子，输出过程，可以用一个全局变量`cnt`记录步数。

AC代码：

```
#include<stdio.h>
int cnt=0;
void Move(int low,int high,char s,char t)
{
    if(low==high)
    {
        printf("step #%d: move %d from %c to %c\n",++cnt,low,s,t);
        return;
    }
    Move(low+1,high,s,'A'+'B'+'C'-s-t); //可以用('A'+'B'+'C'-s-t)代表辅助杆，
    Move(low,low,s,t);                  //当然再传一个char类型参数记辅助杆也可
    Move(low+1,high,'A'+'B'+'C'-s-t,t);
}
int main()
{
    int n;
    scanf("%d",&n);
    Move(1,n,'A','C');
    return 0;
}
```

J. Crazy Cells

解题思路：

可以使用数组a记录每个年龄的细胞数,$a[i]$表示当前年龄为i的细胞数。每次根据现在的各个年龄段的细胞数可以推出下一时刻每个年龄的细胞数，推q次即可。

AC代码：

```
#include <stdio.h>

int x,y,z,q;
long long a[1000],ans;

int main()
{
    scanf("%d%d%d%d",&x,&y,&z,&q);
    q--;
    a[0]=1;
    while(q--)
    {
        for(int i=x+y+z;i>=1;i--) a[i]=a[i-1];
        a[0]=0;
        for(int i=x;i<x+y;i++) a[0]+=a[i];
    }
    for(int i=0;i<x+y+z;i++) ans+=a[i];
    printf("%lld",ans);

    return 0;
}
```

K.飞翔的天使（加强版）

解题思路：

假设天使能够最早在t时刻到达终点，最优的策略必然是在$0-t$的时间内，速度的方向不变且大小最大。（证明：把风吹的路程和提供速度走过的路程分离，提供速度走过的路程两点之间直线最短）

所以只需要找出在某个时间点**纯靠风吹**的位置，并判断能否从这里在t时间内以v速率**直线奔向终点**即可。故我们可以从前往后递推求解，求天使在第i段风的时间里能不能到达、什么时候到达终点。

设在这段风开始之前的总时间为$t[i]$，风总共吹到（lx,ly），这段风结束的时刻为$t[i+1]$，吹到（wx,wy）。

设$dT=t[i+1]-t[i],dx=wx-lx,dy=wy-ly$，在这段风$\lambda\in[0,1]$比例的时刻$t[i]+\lambda dT$到达终点，这时被吹到（$lx+\lambda dx,ty+\lambda dy$）。

列出方程：$\sqrt{(lx+\lambda dx-tx)^2+(ly+\lambda dy-ty)^2}=v_{max}\times(t[i]+\lambda dT)$

解这个一元二次方程即可求出λ，当解位于$[0,1]$区间时是可行解，如果有一个可行解即可直接输出；如果有两个可行解，取较小者输出；无解则继续枚举下一个时间段。当然，要考虑退化到一元一次方程的可能。

AC代码：

```
#include<stdio.h>
#include<math.h>
double sx,sy,tx,ty,v;
#define N 10010
int n,k,t[N];
double res,wix[N],wiy[N];
int jud(double a,double b,double c){
    double t,delta=b*b-4*a*c;
    if(fabs(a)<1e-9){
        double x=-c/b;
        if(x>0&&x<1){
            res=x;
            return 1;
        }
        return 0;
    }
    if(delta<0)return 0;
    double x1=(-b+sqrt(delta))/(2*a),x2=(-b-sqrt(delta))/(2*a);
    if(x1>x2)t=x1,x1=x2,x2=t;
    if(x1>=0&&x1<=1){res=x1;return 1;}
    if(x2>=0&&x2<=1){res=x2;return 1;}
    return 0;
}
int main(){
    int i;
    scanf("%lf%lf%lf%lf%d%d%lf",&sx,&sy,&tx,&ty,&n,&k,&v);
    double lx=sx,ly=sy;
    for(i=1;i<=n;i++)scanf("%d%lf%lf",&t[i],&wix[i],&wiy[i]);
    t[n+1]=k;
    for(i=1;i<=n;i++){
        double dT=t[i+1]-t[i],dx=wix[i]*dT,dy=wiy[i]*dT,wx=lx+dx,wy=ly+dy;
        if(jud(
```

```
            dx*dx+dy*dy-pow(v*dT,2),
            2*(ly-ty)*dy+2*(lx-tx)*dx-2*t[i]*v*v*dT,
            pow(lx-tx,2)+pow(ly-ty,2)-pow(t[i]*v,2))
        ){
            printf("%.10f\n",dT*res+t[i]);
            return 0;
        }
        lx=wx;ly=wy;
    }
}
```