A. 菲菲公主的ReLU (II)

解题思路

依照题意输出即可。

AC代码

```
#include<stdio.h>
#include<math.h>
int main(){
    int x;
    while(~scanf("%d",&x))
        printf("%.2f\n",x>0?x:exp(x)-1);
    return 0;
}
```

B. 铺路

解题思路

用题目中的地砖铺路只可能是每次用两个砖拼一起铺两列，所以如果路的长度n为奇数则无法进行铺路，如果路的长度n为偶数，则两两分组共可分为$n/2$组，每组有两种铺法，共有$2^{\frac{n}{2}}$种铺法。

AC代码

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

int main() {
    int n;
    while (scanf("%d", &n) != EOF) {
        if (n % 2 == 1) {
            printf("0\n");
        }
        else {
            printf("%d\n", (int) pow(2, n / 2));
        }
    }
    return 0;
}
```

C. Rerevse

解题思路

本题需要对字符串s的一部分进行反转操作，所以我们设置两个变量l和r，分别表示待反转字符串的左端和右端，每次交换$s[l]$和$s[r]$，并令$l=l+1$，$r=r-1$，直到$l \geq r$的时候结束。

AC代码

```c
#include <stdio.h>

int main() {
    char s[10010], temp;
    int l, r;
    scanf("%s", s);
    scanf("%d%d", &l, &r);
    for (l--, r--; l < r; l++, r--) {
        temp = s[l];
        s[l] = s[r];
        s[r] = temp;
    }
    printf("%s", s);
    return 0;
}
```

D. 幸运数字

解题思路

依照$Hint$里的代码进行二分即可。这里提供了一种和提示中等价的写法。

二分的时候一定要注意上界和下界，以及这两个界分别具有的含义。比如这个题里l的含义应该是**已经确定的**、对应元素**小于等于**x的最大下标，r的含义是**已经确定的**、对应元素**大于等于**x的最小下标。在其他的题目中，一定也要注意判断条件是“小于”还是“小于等于”，否则很容易出现逻辑混乱。

这里提供一个去年写过的$blog$，仅供参考：这里

AC代码

```
#include<stdio.h>
#define N 1000010
int a[N];
int main(){
    int i,n,l=0,r,x,ans=0;
    scanf("%d",&n);
    r=n-1;
    for(i=0;i<n;i++)scanf("%d",&a[i]);
    scanf("%d",&x);
    while(l<r-1){
        int mid=(l+r)>>1;
        ++ans;
        if(a[mid]==x)return printf("YES\n%d",ans),0;
        else if(a[mid]>x)r=mid;
        else l=mid;
    }
    printf("NO\n%d %d",a[l],a[r]);
    return 0;
}
```

E. 不TLE的平均值

解题思路

本题题意是要我们设计一种查询极快的查询算法，因为总数据量很大，查询次数也很多，如果每次查询都遍历区间里所有数字的话肯定会TLE。

我们采用“缓存”的思想，即在读入n个数字的时候就计算好部分数据并存起来，在查询的时候直接取出来用就好，这种思想实际上在用空间换时间，存的数据越多，时间花费越少。即俗话说的“好记性不如烂笔头”。

在本题中，我们存储的数据是前n个数的和。查询第L个数字到第R个数字的平均值的时候，使用前R个数字的和减去前$(L-1)$个数字的和再除以$(R-L+1)$即可得答案，不用遍历所有$(R-L+1)$个数字，查询效率很高。

AC代码

```c
#include <stdio.h>

long long sum[1000010];

int main() {
    int m, n, i, g, l, r;
    scanf("%d%d", &m, &n);
    for (i = 1; i <= n; i++) {
        scanf("%d", &g);
        sum[i] = sum[i - 1] + g;
    }
    for (i = 0; i < m; i++) {
        scanf("%d%d", &l, &r);
        printf("%d\n", (sum[r] - sum[l - 1]) / (r - l + 1));
    }
    return 0;
}
```

F. 整理成绩

解题思路

在读入后，首先使用双关键字排序按要求排好序，具体实现方法见AC代码以及$C6$题解，不再赘述。在输出的时候，如果当前学号在前面已经输出过了，则跳过，否则输出当前学号和成绩。

AC代码

```c
#include <stdio.h>

int main() {
    int grade[2019], id[2019];
    int i, j, n = 0, temp, flag;
    while (scanf("%d %d", &id[n], &grade[n]) != EOF) {
        n++;
    }
    for (i = 0; i < n; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (grade[j]<grade[j + 1] || (grade[j]==grade[j + 1] &&
id[j]>id[j + 1])) {
                temp = id[j];
                id[j] = id[j + 1];
                id[j + 1] = temp;
                temp = grade[j];
                grade[j] = grade[j + 1];
                grade[j + 1] = temp;
            }
        }
    }
    for (i = 0; i < n; i++) {
        flag = 0;
        for (j = 0; j < i; j++) {
            if (id[j] == id[i]) {
                flag = 1;
                break;
            }
        }
        if (flag == 0) {
            printf("%d %d\n", id[i], grade[i]);
        }
    }
    return 0;
}
```

AC代码2

也可以像$hint$中说的那样，用一个数组进行标记，标记方法有很多，大家可任选。

（参考代码来自19373718 周勤）

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>
#include<ctype.h>
int a[2005][2];
char b[100000000]={0};
int main()
{
    int i=0,j,k,temp;
    while(~scanf("%d%d",&a[i][0],&a[i][1])) i++;
    for(j=0;j<i-1;j++){
        for(k=0;k<i-j-1;k++){
            if(a[k][1]<a[k+1][1]){
                temp=a[k+1][1];
                a[k+1][1]=a[k][1];
                a[k][1]=temp;
                temp=a[k+1][0];
                a[k+1][0]=a[k][0];
                a[k][0]=temp;
            }
            else if(a[k][1]==a[k+1][1]&&a[k][0]>a[k+1][0]){
                temp=a[k+1][1];
                a[k+1][1]=a[k][1];
                a[k][1]=temp;
                temp=a[k+1][0];
                a[k+1][0]=a[k][0];
                a[k][0]=temp;
            }
        }
    }
    for(j=0;j<i;j++){
        if(!b[a[j][0]]){
            b[a[j][0]]=1;
            printf("%d %d\n",a[j][0],a[j][1]);
        }
    }
    return 0;
}
```

G. 斐波那契数列(PLUS)

解题思路

大家都会递推求斐波那契数列和取模，在这里就不说了。

难点在于对于答案x分解质因数：

假设$x = p_1^{k_1} p_2^{k_2} \dots p_n^{k_n}$，其中$p_1 < p_2 < \dots < p_n$。

从2开始枚举x的质因数p_i，如果是x的因数那么将x不停除以这个数（让x变为$p_{i+1}^{k_{i+1}} \dots p_n^{k_n}$，以便枚举下一个质因数$p_{i+1}$）。当当前枚举的数字的平方超过$x$的当前值时，跳出循环结束判断。此时，如果$x$大于1，那么它一定为质数，否则已经分解质因数完毕。可以手造几组x来思考为什么是这样子。

AC代码

```
#include<stdio.h>
long long f[50]={0,1};
int main(){
    int i,n,x,flag=0;
    scanf("%d",&n);
    for(i=2;i<=n;i++)f[i]=(f[i-1]+f[i-2])%(1u<<31);
    printf("%d=",x=f[n]);
    for(i=2;1LL*i*i<=x;i++){
        while(x%i==0){
            if(flag)printf("*");
            printf("%d",i);
            flag=1;
            x/=i;
        }
    }
    if(x>1){
        if(flag)printf("*");
        printf("%d",x);
    }
    return 0;
}
```

H.旋转吧！字符串

解题思路

本题主要考察字符串相关的操作，大体的思路按照题目的意思来即可，当然如果希望处理的时候方便一些，可以考虑把字符串数组开大一点，然后把读入的字符串存到数组的中间再进行处理，但大体思路仍是相似的。

本题的一个实现技巧是可以开一个临时数组，将原始字符串中的非空字符复制过来，再通过strcpy()把临时数组复制给原数组，达到去除'\0'的目的。

此外，本题需要注意的一点是字符串长度每次旋转后可能会改变，需要及时更新。

AC代码

```
#include <stdio.h>
#include <string.h>
char s[300];
int len;
void rotate(int c,int r){
    int i;
    for (i=1;i<=r;i++){
        if (c-i>=0&&c+i<len){
            char tmp = s[c+i];
            s[c+i] = s[c-i];
            s[c-i] = tmp;
        }
        else if (c-i>=0){
            s[c-i]='\0';
        }
        else if (c+i<len){
            s[c+i] = '\0';
        }
    }
    char temp[300]={0};
    int a=0;
    for (i=0;i<len;i++){
        if (s[i]) temp[a++] = s[i];
    }
    strcpy(s,temp);
    len = strlen(s);
}
int main(){
    int n;
    scanf("%s",s);
    len = strlen(s);
    scanf("%d",&n);
    while(n--){
        int c,r;
        scanf("%d %d",&c,&r);
        rotate(c,r);
    }
    printf("%s",s);
}
```

I.表达式输出

解题思路

本题主要考察大家对递归的理解和运用。一般来说，遇到递归的题目，最重要的是找到每次递归终止的**边界条件**以及不同情况之间的转换关系（也就是**递归方程**）。

以本题为例，这道题要求将每一对()内的元素输出2次，显然由于()的嵌套性，可以通过递归来解决（当然也可以用其他方法，比如**栈**）。对于每一层的递归，它的递归返回条件是遇到')'或者'\0'；继续向下一层的条件是遇到又一个'('；而对于其他的情况，只要正常地进行输出即可。具体的实现详见参考代码。

此外，该参考代码的注释语句还提供了一种对递归函数的可行的调试方法，即把每一层具体的调用情况打印到标准输出中，同学们可以尝试一下，相信对于理解标程的实现过程有较大的帮助。

AC代码

```
#include<stdio.h>
char str[101] = {};
int prt(int pos){
    int nextAddr;
    switch(str[pos]){
        case ')':
            //printf(") pos=%d\n",pos);
            return pos + 1;//返回下一轮匹配的开始地址
        case '\0':
            return pos;
        case '(':
            //printf("( pos=%d\n",pos);
            nextAddr=prt(pos + 1);//获取下一轮匹配的开始地址（遇到')'时返回该地址）
            prt(pos + 1);//重复2次用于输出
            return prt(nextAddr);//从nextAddr处继续执行，完毕后返回
        default:
            putchar(str[pos]);
            //putchar('\n');
            //printf("str[pos]=%c pos=%d\n",str[pos],pos);
            return prt(pos + 1);
            break;
    }
}
int main(){
    fgets(str,100,stdin);
    prt(0);
}

```

J. ><

解题思路

如果答案最优，某个x必然向左或向右延伸为依次减一形成的序列，比如`<<>>><`对应`0132101`，3向右延伸到0。

从左到右递推求出来满足连续小于号的情况下的各点最小值，再从右到左递推求出来满足连续大于号的情况下的个点最小值，即为所求。

AC代码

```
#include<stdio.h>
#include<string.h>
#define N 1000010
char a[N];
int x[N];
int main(){
    int i,n;
    while(~scanf("%s",a)){
        long long ans=0;
        n=strlen(a);
        x[0]=0;
        for(i=0;i<n;i++){
            if(a[i]=='<')x[i+1]=x[i]+1;
            else x[i+1]=0;
        }
        for(i=n-1;i>=0;i--)
            if(a[i]=='>'&&x[i]<x[i+1]+1)
                x[i]=x[i+1]+1;
        for(i=0;i<=n;i++)ans+=x[i];
        printf("%lld\n",ans);
    }
    return 0;
}
```

K.遍历迷宫

解题思路

本题是一个简单的模拟题，之所以这样说，是因为机器人的行动方向是十分固定的：它只能向前或者右转。这也就意味着机器人的合理运动轨迹是顺时针地一圈一圈不断向迷宫中心移动，只要该过程中出现障碍，那么机器人所绕的这个圈就会变小，圈外的那些点它也就去不了了，最后肯定会失败。

因此，我们只需要依题意，开一个数组记录机器人每次移动的方向，如果直行遇到障碍就右转，直到无法继续移动为止。注意别忘了**把已经走过的点标记**，以避免出现死循环。

当然本题也有使用**深度优先搜索**的方法，欢迎感兴趣的同学自行探索。

AC代码

```
#include <stdio.h>
#include <string.h>
int dx[4]={0,1,0,-1};
int dy[4]={1,0,-1,0};
int laby[300010];
int dir[300010];

int isOnTable(int x,int y,int n,int m){
    return (x>=0 && x<n && y>=0 && y<m);
}


int main(){
    int t;
    scanf("%d",&t);
    while(t--){
        memset(laby,0,sizeof(laby));
        memset(dir,0,sizeof(dir));
        int n,m,k;
        int x,y;
        scanf("%d %d %d",&n,&m,&k);
        for (int i=0;i<k;i++){
            scanf("%d %d",&x,&y);
            x-=1;y-=1;
            laby[m*x+y] = 1;
        }
        laby[0]=1;
        int curx=0,cury=0;
        int lastx=0,lasty=0;
        int cnt=0,nav=0;
        while(1){
            //printf("%d %d\n",curx,cury);
            curx = lastx+dx[nav];
            cury = lasty+dy[nav];
            if (isOnTable(curx,cury,n,m)&&!laby[curx*m+cury]){//直行
                dir[cnt++] = 0;
            }
            else{
                nav = (nav+1)%4;
                curx = lastx+dx[nav];
                cury = lasty+dy[nav];
```

```
                if (isOnTable(curx,cury,n,m)&&!laby[curx*m+cury]){
                    dir[cnt++] = 1;//转向
                }
                else break;//结束
            }
            lastx = curx;
            lasty = cury;
            laby[lastx*m+lasty]=1;
        }
        //printf("%d\n",cnt);
        if (cnt==n*m-k-1){
            printf("WIN\n");
            for (int i=0;i<cnt;++i){
                if (!dir[i]){
                    printf("straight\n");
                }
                else{
                    printf("turn\n");
                }
            }
        }
        else{
            printf("LOSE\n");
        }
    }
}
```