

前6题都是课件题，在课件代码上稍作修改即可通过，故这里不给出题目分析，只给出参考代码

## A 看比赛简介!!!

```
#include <stdio.h>
int main()
{
    int a,b;
    scanf("%d %d",&a,&b);
    printf("%d",a+b);
}
```

## B 课件题-求余

```
#include <stdio.h>
int main()
{
    int a,b;
    scanf("%d %d",&a,&b);
    printf("%d mod %d = %d",b,a,b%a);
}
```

## C 课件题-判断成绩

```
#include <stdio.h>
int main()
{
    int a;
    scanf("%d",&a);
    printf(a==100?"WellDone!":"KeepGoing!");
}
```

## D 课件题-打印星星

```
#include <stdio.h>
int main()
{
    int n;
    scanf("%d", &n);
    n++;
    while (n-->0)
        putchar('#');
    return 0;
}
```

## E 课件题-简单平均分

```
#include <stdio.h>
int main()
{
    int sum=0,n=0;
    int score;
    scanf("%d",&score);
    while(score>=0 && score<=100)
    {
        sum+=score;
        n++;
        scanf("%d",&score);
    }
    printf("%d\n",sum/n);

    return 0;
}
```

## F 课件题-最小公倍数

```
#include <stdio.h>
int main()
{
    int a,b;
    scanf("%d%d",&a,&b);
    int lcm=a;
    if(lcm<b)lcm=b;

    while(!(lcm%a==0&&lcm%b==0))lcm++;

    printf("%d\n",lcm);

    return 0;
}
// 这份代码效率更高
#include <stdio.h>
int main()
{
    int a, b, tmp, gcd;
    scanf("%d %d", &a, &b);
    int m = a, n = b;
    if (a < b)
    {
        tmp = a;
        a = b;
        b = tmp;
    }
```

```

}
while (b > 0)
{
    gcd = a % b;
    a = b;
    b = gcd;
}
printf("%d", m * n / a);
}

```

G dwy考美院

难度	考点
3	字符画

## 题目分析

本题是简单的字符画题，将输出样例中的所有 `\` 替换为 `\\`，之后再加入 `\n` 即可。

## 示例代码

[illegible]

[illegible]

## H 击败爆炎树

难度	考点
3	简单数学运算，数据范围

## 题目分析

本题题面并不难懂，即对旅行者拥有的每一个技能进行检测判断是否能够对boss造成足够的伤害

思路：

1. 读取法力值 $k$ ，技能个数 $m$ ，BOSS的体力值 $n$ ，循环读取 $m$ 个技能的消耗的法力值 $a[i]$ ，伤害值 $b[i]$ 。
2. 遍历每个技能，进行判断。

若技能的伤害值 $b[i] == 0$ ，即没有伤害，跳过该技能；

若技能的伤害值 $b[i] \neq 0$ 且消耗的法力值 $a[i] == 0$ ，即可以无限释放技能， $flag = 1$ 表示能杀死BOSS，输出该技能的编号 $i + 1$ （ $i$ 是从0开始的）；

若技能的伤害值 $b[i] \neq 0$ , 消耗的法力值 $a[i] \neq 0$ , 且能释放技能的个数 (法力值 $k$ /该技能消耗的法力值 $a[i]$ ) \* 该技能的伤害值 $b[i] \geq$ BOSS的体力值 $n$ , 即能杀死BOSS, 输出该技能的编号 $i + 1$ 。

- 3.判断是否能击败boss，若不能，遍历每一个技能，输出造成的最大伤害。

## 示例代码

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

#define LL long long //注意虽然给出数据n,m,k,a,b的范围都在int范围内
//但经过乘法计算后很可能超出int范围导致出错,所以采用long long类型

int main()
{
    LL i;
    LL k;           //法力值
    LL m;           //技能个数
    LL n;           // BOSS的体力值
    LL a[30010];    //每个技能耗费的法力值
    LL b[30010];    //每个技能造成的伤害
    LL yon = -1;    //是否能杀死BOSS, -1不能, 1能
```

```

LL maxn = 0; //不能击败boss时能造成的最大伤害
scanf("%lld %lld %lld", &k, &m, &n);
for (i = 0; i < m; i++)
{
    scanf("%lld %lld", &a[i], &b[i]);
}
for (i = 0; i < m; i++)
{
    if (a[i] == 0 && b[i] != 0) //要特别注意法力值消耗或技能伤害值为零的情况
    {
        printf("%lld ", i + 1); //当法力值消耗为0，伤害为正数，可以无限放技能，肯定能打败
boss
        yon = 1;
    }
    else if (a[i] == 0 && b[i] == 0) // 0魔0攻的技能无意义，直接跳过。
    {
        continue;
    }
    else if (a[i] != 0 && b[i] == 0) //耗魔0攻的技能同样无意义
    {
        continue;
    }
    else if ((k / a[i]) * b[i] >= n) //注意，如果前面不对法力值消耗为0的技能进行预先处理
        //在这里就会出现除以0的情况，导致运行时出错
    {
        printf("%lld ", i + 1);
        yon = 1; //判定为能够击败boss
    }
}
if (yon == -1) //如果不能击败boss
{
    for (i = 0; i < m; i++)
    {
        if (a[i] < k)
        {
            if (a[i] == 0 && b[i] == 0)
                continue; //同上述处理。
            int step = (k / a[i]) * b[i];
            if (step > maxn)
                maxn = step;
        }
    }
    printf("%lld", maxn);
}
return 0;
}

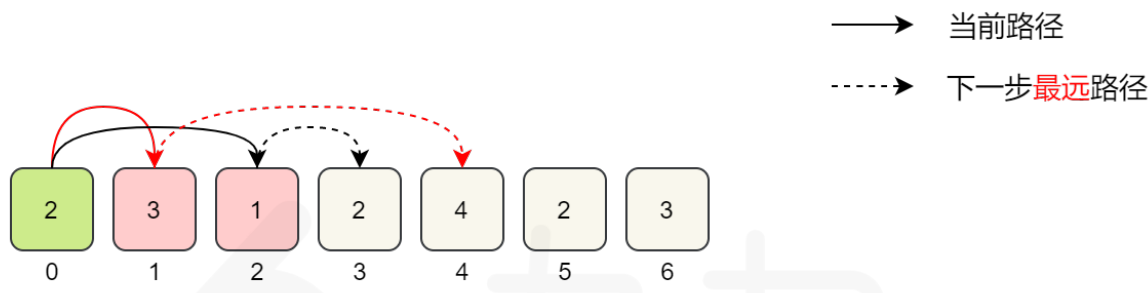
```

# I MRIYA我们回家了

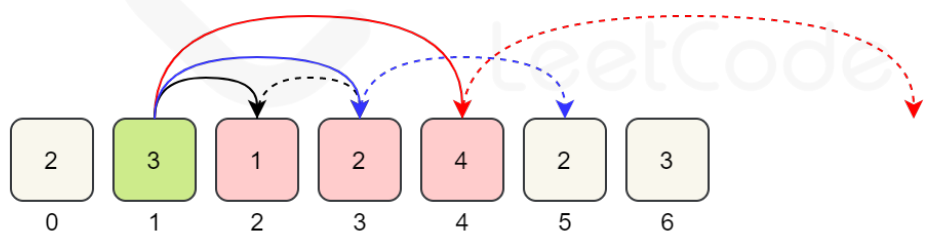
# 题目分析

本题由经典贪心算法“最远跳跃距离”改编而来,通过局部最优得出全局最优。

对于每一次传送，我们查找所有可传送范围内的传送点，并将可传送到距离起点最远位置的传送点作为下一次传送的起点，直到到达终点。在具体代码实现中，我们引入两个标记f1和f2，其中f1表示当前可传送范围内所有点中最远可传送到的位置，f2表示当前的最远传送距离。当扫描到f2点时，进行下一次传送，并用f1替换掉f2。



从下标 0 出发，可以跳到下标 1 和下标 2，下标 1 可以跳得更远，选择下标 1



从下标 1 出发，可以跳到下标 2、3 和 4，下标 4 可以跳得更远，选择下标 4

# 实例代码

```
#include <stdio.h>
int main()
{
    int t;
    scanf("%d", &t);
    while (t-->0)
    {
        int n;
        int ans = 0;
        int f1 = 0;
        int f2 = 0;
        scanf("%d", &n);
        for (int i = 0; i < n; i++)
        {
            int k;
```

```

scanf("%d", &k);
if (i == n - 1)
{
    break;
}
if (i + k > f1)
{
    f1 = i + k;
}
if (i == f2)
{
    f2 = f1;
    ans++;
}
}
printf("%d\n", ans);
}
return 0;
}

```

## J 呱呱泡蛙的数学卡片

这里需要对样例和题面进行分析。

首先，根据“卡片上的内容全是正整数”、“ $tsu \times tsu = tsu$ ”，得知“ $tsu$ ”是指1。

“ $ketsu \times ketsu = keketsu$ ”，长得很像含有1的东西的乘法。

“ $tsu + ketsu = tsuketsu$ ”，加法的1加到了最左边。

“ $tsu + roma = ketsu$ ”，因此，roma、ketsu、tsuketsu是相邻正整数，能解释的只有“ketsu”是“10”，roma是前一个数，“tsuketsu”是“11”。

“ $tsuketsu \times tsuketsu = tsukemaketsu$ ”，这一行是典型的“ $11 \times 11 = 121$ ”，所以“ma”是2。

“ $ma \times ma = rotsu$ ”，那么“rotsu”是4。

“呱呱泡蛙的计数系统和人类不太一样”、“呱呱泡蛙有两只手，一只手有三个手指”，暗示了进制不太一样。

“ $ma \times rotsu = maketsu$ ”、“ $rotsu \times rotsu = rotsukema$ ”，2乘4得12，能说明是6进制。在6进制，4乘4是24。

另外的入手点是试样例，也可能发现答案是6进制，6的3次方是216，和上面三个值222、223、252相近。

至此问题解决。这些数是从右往左写的，并且，数位间隔用“ke”，0不写，1是tsu，2是ma，3是ro，4是rotsu，5是roma。

因此10（6进制的10是10进制的6）是ketsu，100（6进制的100是10进制的36）是keketsu，101（6进制的101是10进制的37）是tsukeketsu，110（6进制的110是10进制的42）是ketsuketsu，111（6进制的111是10进制的43）是tsuketsuketsu，以此类推。

经过以上的推理，不难写出程序：

```
#include<stdio.h>
int main()
{
    int a;
    while(~scanf("%d",&a))
    {
        while(a!=0)
        {
            if(a%6==1)
            {
                printf("tsu");
            }
            else if(a%6==2)
            {
                printf("ma");
            }
            else if(a%6==3)
            {
                printf("ro");
            }
            else if(a%6==4)
            {
                printf("rotsu");
            }
            else if(a%6==5)
            {
                printf("roma");
            }
            a/=6;
            if(a!=0)
            {
                printf("ke");
            }
        }
        printf("\n");
    }
}
```

## K 雀魂绝艺总纲-基础篇

### 题目解析

搜索刻子或顺子，直到剩下两张牌，再判断这两张牌是否是对子。



## 示例代码

```
#include <stdio.h>
#include <string.h>
int msp[3][10]; // 0m1s2p
int check(int remain)
{
    if (remain == 2)
    {
        // check DD
        for (int i = 0; i <= 9; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                if (msp[j][i] == 2)
                {
                    return 1;
                }
            }
        }
        return 0;
    }

    // check ABC
    for (int i = 1; i <= 7; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            if (msp[j][i] > 0 && msp[j][i + 1] > 0 && msp[j][i + 2] > 0)
            {
                msp[j][i]--;
                msp[j][i + 1]--;
                msp[j][i + 2]--;
                if (check(remain - 3))
                {
                    return 1;
                }
                msp[j][i]++;
                msp[j][i + 1]++;
                msp[j][i + 2]++;
            }
        }
    }

    // check KKK
    for (int i = 1; i <= 9; i++)
    {
```

```

    for (int j = 0; j < 3; j++)
    {
        if (msp[j][i] >= 3)
        {
            msp[j][i] -= 3;
            if (check(remain - 3))
            {
                return 1;
            }
            msp[j][i] += 3;
        }
    }
}

return 0;
}

int main(int argc, const char *argv[])
{

    int n, l;
    char c, temp[330];
    while (fgets(temp, sizeof(temp), stdin) != NULL)
    {
        memset(msp, 0, sizeof(msp));

        l = (int)strlen(temp);
        if (temp[l] == '\n')
        {
            l -= 1;
        }
        if (l / 2 != 14)
        {
            printf("no-ten...\n");
            continue;
        }

        for (int i = 0; i < 14; i++)
        {
            sscanf(temp + i * 2, "%d%c", &n, &c);
            switch (c)
            {
                case 'm':
                    msp[0][n]++;
                    break;
                case 's':
                    msp[1][n]++;
                    break;
                case 'p':
                    msp[2][n]++;

```

```
        break;
    default:
        break;
    }
}

if (check(14))
{
    printf("ron!\n");
}
else
{
    printf("no-ten...\n");
}

return 0;
}
```