T1

给出宝塔的层数n，和每层的高度k，画出宝塔。

1. 宝塔一共有n层，每一层的最窄处等于上一层的最宽处。
2. 每层宝塔由“塔顶”和“塔身”组成。每层塔顶和塔身高度相等，即都由k层砖块搭成。
3. 塔顶每层砖块的宽度逐渐递增，宝塔的第一层塔顶的第一层砖块宽度为1。
4. 塔身的宽度等于该层塔顶倒数第2层砖块宽度。
5. 最后，这个塔底有一个地基，地基的宽度为整个塔身最宽处的宽度。

输入：
第一个数为塔的层数n，第二个数为每层的高度k。
输出：
画出的宝塔

输入示例： 3 3

输出示例：

```
            *
          *   *
        *********
          *   *
          *   *
          *   *
        *********
      *           *
    *****************
      *           *
      *           *
      *           *
    *****************
  *                   *
*************************
  *                   *
  *                   *
  *                   *
*************************
```

T2

教材上介绍了链表的常见形式有单向循环链表。当然，链表尾节点的指针可以指向链表中的任意节点，这样的链表我们称为“有环”。
题目给出的预置代码，给定链表初始化函数和主函数，请仔细阅读。
请编写函数 Detection( ) 判断程序创建的单向链表是否“有环”。

预置代码：

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

```
//构造结构体
typedef struct list
{
    char data;
    struct list *next;
}*List,LNode;

//函数声明
List init_list(List head);
int Detection(List head);
int main()
{
    List head;
    head = (LNode*)malloc(sizeof(LNode));
    head->next = NULL;
    head = init_list(head);
    if(Detection(head))
        printf("单链表中有环!\n");
    else
        printf("单链表中无环!\n");
}

//链表初始化函数
List init_list(List head)
{
    int i = 0;
    List p = head;
    char chain[10000];
    gets(chain);
    while(chain[i]!='\0')
    {
        if(chain[i]=='@')
        {
            int link=chain[++i]-'0';
            List q=head;
            while(link--)
                q=q->next;
            p->next = q->next;
            break;
        }
        List s;
        s = (LNode*)malloc(sizeof(LNode));
        s->data = chain[i];
        s->next = NULL;
        p->next = s;
        p = p->next;
        i++;
    }
    return head;
}
```

在这里分享下我的答案：

```
int Detection(List head)
{
```

```
    int sum=0;
    while(head)
    {
        head=head->next;
        sum++;
        if(sum==3000)
        {
            break;
        }
    }
    if(sum==3000)
    {
        return !NULL;
    }
    else
    {
        return NULL;
    }
}
```

T3

起点(0,0)，要求每步只能沿着向上、下、左、右任意一个方向走一格长度。假设根据实际情况要求，存在一个点不能通过。

给定输入终点(m,n)坐标和不能通过的点(p,q)，其中m,n,p,q都为正整数，点(p,q)不是起点也不是终点，且0<p<m，0<q<n。

求从点(0,0)出发不经过点(p,q)到点(m,n)的最短路径的条数，并输出在所有最短路径中所经过点的纵坐标和最小的那一条路径。

比如，在下图从点(0,0)到点(2,1)的所有路径中，点(0,1)不能通过，最短的路径有两条：

(0,0)->(1,0)->(2,0)->(2,1)和(0,0)->(1,0)->(1,1)->(2,1)，其中所经过的点的纵坐标的和最小的是(0,0)->(1,0)->(2,0)->(2,1)这条。

程序的输入有两行：

第一行为终点坐标；

第二行为不能通过的点的坐标（输入-1 -1表示所有点都可以通过）

输出也有两行：

第一行为最短路径的条数；

第二行为在所有最短路径中，纵坐标的和最小的那一条路径（没有路径可以达到终点输出：“No path!”）。

```
示例1：
输入：
2 1
1 0
输出：
1
(0,0)(0,1)(1,1)(2,1)
```

示例2：
输入：
```
2 1
-1 -1
```
输出：
```
3
(0,0)(1,0)(2,0)(2,1)
```

T4

现在有一串经过加密过后的字符串，需要解密。解密过程为：逐层翻转括号里的字符。

例如，对于字符串 ： (em(gr(ma))orp)
先翻转最里面的括号 gr(ma)->gram，然后再翻转 em(gram)orp->emmargorp，最后翻转最外面的括号 (emmargorp)->programme

输入：

一串加密过的字符串。

输出：

经过逐层翻转括号后得到的字符串。