1. 求最大公约数

==> n m r 移位

int ldd(int n,int m){

int r = 0;

while (m>0){

r = n%m;

n = m;

m = r;

}

return n;

}

#### [合并两个有序链表](https://leetcode-cn.com/problems/merge-two-sorted-lists/)

#### 将两个升序链表合并为一个新的升序链表并返回。新链表是通过拼接给定的两个链表的所有节点组成的。

****示例：****

****输入：****1->2->4, 1->3->4****输出：****1->1->2->3->4->4

/\*\*

 \* Definition for singly-linked list.

 \* struct ListNode {

 \*     int val;

 \*     struct ListNode \*next;

 \* };

 \*/

struct ListNode\* mergeTwoLists(struct ListNode\* l1, struct ListNode\* l2){

    if(l1 == NULL){

        return l2;

    }

    if(l2 == NULL){

        return l1;

    }

    typedef struct ListNode ListNode;

    ListNode\* newH = (ListNode\*)malloc(sizeof(ListNode));

    ListNode\* H = newH;

    while(l1 != NULL && l2 != NULL){

        if(l1->val < l2->val){

            newH->next = l1;

            l1=l1->next;

        }else{

            newH->next = l2;

            l2 = l2->next;

        }

        newH=newH->next;

    }

    if( l1 != NULL){

        newH->next = l1;

    }

    if(l2 != NULL){

        newH->next = l2;

    }

    return H->next;

}

3、

编写代码，以给定值x为基准将链表分割成两部分，所有小于x的结点排在大于或等于x的结点之前

给定一个链表的头指针 ListNode\* **pHead**，请返回重新排列后的链表的头指针。注意：分割以后保持原来的数据顺序不变。

/\*

struct ListNode {

int val;

struct ListNode \*next;

ListNode(int x) : val(x), next(NULL) {}

};\*/

class Partition {

public:

ListNode\* partition(ListNode\* pHead, int x) {

if(NULL == pHead){

return NULL;

}

ListNode \*L,\*H,\*LN,\*HN,\*cur;

LN=L=(ListNode\*)malloc(sizeof(ListNode));

HN=H=(ListNode\*)malloc(sizeof(ListNode));

cur = pHead;

while(cur){

if(cur->val < x){

L->next = cur;

L = L->next;

}else{

H->next = cur;

H = H->next;

}

cur = cur->next;

}

H->next = NULL;

L->next = HN->next;

return LN->next;

}

};

// 先实现栈

typedef char Type;

typedef struct{

    Type\* \_array;

    size\_t \_top;

    size\_t \_capacity;

}Stack;

// 初始化空栈

void stackInit(Stack\* s){

    // s = (Stack\*)malloc(sizeof(Stack));

    s->\_capacity = 8;

    s->\_array = (Type\*)malloc(sizeof(Type)\*s->\_capacity);

    s->\_top = -1;

}

// push

void pushBack(Stack\* s,Type data){

    if( NULL == s){

        return ;

    }

    if( s->\_top + 1 == s->\_capacity){

        s->\_capacity \*=2;

        s->\_array = (Type\*)realloc(s->\_array,s->\_capacity\*sizeof(Type));

    }

    s->\_array[++s->\_top] = data;

}

// pop

void popBack(Stack\* s){

    if(s->\_top == -1){

        return ;

    }

    s->\_top--;

}

// top

Type stackTop(Stack\* s){

    // if(-1 == s->\_top){

    //     return (Type)0;

    // }

    return s->\_array[s->\_top];

}

// 判空

int stackEmpty(Stack\* s){

    if(s->\_top == -1)

        return 1;

    return 0;

}

bool isValid(char \* s){

    Stack st;

    stackInit(&st);

    // 映射匹配

    static char map[][2]={{'(',')'},{'[',']'},{'{','}'}};

    while(\*s){

        int i =0;

        // 映射左半入栈

        for(;i<3;i++){

            if( \*s == map[i][0]){

                pushBack(&st,\*s);

                ++s;

                break;

            }

        }

        // 如果不是左半边就是右半情况

        if(3 == i){

            if(stackEmpty(&st)){

                return false;

            }

            char ctop = stackTop(&st);

            for(int j = 0 ;j<3;j++){

                if(\*s == map[j][1]){

                    if(ctop == map[j][0]){

                        popBack(&st);

                        s++;

                        break;

                    }else{

                        return false;

                    }

                }

            }

        }

    }

    if(stackEmpty(&st)){

        return true;

    }

    return false;

}

给定一个只包括 '('，')'，'{'，'}'，'['，']' 的字符串，判断字符串是否有效。

有效字符串需满足：

左括号必须用相同类型的右括号闭合。

左括号必须以正确的顺序闭合。

注意空字符串可被认为是有效字符串。

示例 1:

输入: "()"

输出: true

示例 2:

输入: "()[]{}"

输出: true

示例 3:

输入: "(]"

输出: false

示例 4:

输入: "([)]"

输出: false

示例 5:

输入: "{[]}"

输出: true