翻转单词顺序

输入一个英文句子，翻转句子中单词的顺序，但单词内字符的顺序不变。为简单起见，标点符号和普通字母一样处理。例如输入字符串"I am a student. "，则输出"student. a am I"。

示例 1：

输入："the sky is blue"

输出："blue is sky the"

示例 2：

输入：" hello world! "

输出："world! hello"

解释：输入字符串可以在前面或者后面包含多余的空格，但是反转后的字符不能包括。

示例 3：

输入："a good   example"

输出："example good a"

解释：如果两个单词间有多余的空格，将反转后单词间的空格减少到只含一个。

```
 /*void reverse(char* left,char* right)
    {
        while(left<right)
        {
            int temp=*left;
            *left=*right;
            *right=temp;
            left++;
            right--;
        }
    }
char* reverseWords(char* s){
    int n=strlen(s);
    int t=0;
    char* left=s;
    char* right=s+n-1;
    while(left=' ')
    {
        left++;
    }
    while(right==' ')
    {
```

```
        right--;
    }

    while(*s)
    {
        char* start=s;
        while((*s != ' ')&&(*s !='\0'))
        {
            s++;
        }
        char* end=s-1;
        reverse(start,end);
        while(*s==' ')
        {
            s++;
        }
    }
    reverse(left,right);
    return left;
}*/
int lenth(char *s){
    int len = 0;
    if(s[0] == '\0') return 0;
    while(s[++len] != '\0');
    return len;
}
char* reverseWords(char* s){

    int len = lenth(s);

    if(!len || !s) return "";//空字符串的情况
    //双指针一个用于找空格，一个固定单词边界
    int p = 0,q = len-1,head;
    char *ans = (char*)malloc(sizeof(char)*(len+1));
    while(s[p] == ' ' && p<=q) p++;
    while(s[q] == ' '&& p<=q) q--;//去首尾空格
    if(p > q) return "";//只有空格无单词

    head = p;//记录头指针
    p = q;
    int i,j = 0;
    while(1){
        while(p >= 0 && s[p] != ' ') p--;//p 移动至要遍历的单词的左边
```

```
        for(i = p+1;i <= q;i++){//单词赋值给 ans
            ans[j++] = s[i];
        }
        if(p+1 == head){//最后一个单词无需加空格
            break;
        }
        ans[j++] = ' ';
       while(s[p] == ' ') p--;//越过两个单词间的空格
        q = p;//前往下一个单词
    }
    ans[j] = '\0';

   return ans;
}
```

c 测试（简单无空格）

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

void reverse_word(char* first,char* end)
{
    //char* first=string;
    //char* end=string+strlen(string)-1;
    while(first<end)
    {
        int temp=*first;
        *first=*end;
        *end=temp;
        first++;
        end--;
    }
}
void reverse_string(char* str)
{
    char* left=str;
    char* right=str+strlen(str)-1;

    while(*str)
    {
        //printf("***************");
        char* start=str;
        //char* end=str;
```

```
            while((*str !=' ')&&(*str != '\0'))
            {
                  str++;
            }

            reverse_word(start,str-1);
            if(*str == ' ')
            {
                  str++;
            }
      }
      reverse_word(left,right);//整个字符串逆转
}

/*void reverse_string(char* str)
{
      char* left=str;
      char* right=str+strlen(str)-1;
      reverse_word(left,right);//整个字符串逆转
      while(*str)
      {
            char* start=str;
            while((*str !=' ')&&(*str != '\0'))
            {
                  str++;
            }
            char* end=str-1;
            reverse_word(start,end);
            if(*str==' ')
            {
                  str++;
            }
      }
}*/
int main()
{
      char arr[100]={0};
      gets(arr);
      reverse_string(arr);
      printf("%s",arr);
      return 0;
}
```