一、下面两段程序运行结果有何不同，为什么？

#include <stdio.h>

void fun(int a,int b)

{ int t ;

t=a;a=b;b=t;

}

void main()

{ int x=1,y=3;

fun(x,y);

printf("%d，%d",x,y);

}

#include <stdio.h>

void fun(int \*a,int \*b)

{ int t ;

t=\*a;\*a=\*b;\*b=t;

}

void main()

{ int x=1,y=3;

fun(&x,&y);

printf("%d，%d",x,y);

}

两、下面程序运行后结果是什么，为什么？

#include <stdio.h>

static int a=2;

int sum()

{

static int b = 3;

int c = 0;

b += 2;

c += 1;

return(a + b + c);

}

int main(void)

{

int i;

for(i = 0; i < 3; i ++)

{

sum();

}

return 0;

}

二、编程题

字符串操作器

编写一个 C 程序，其中包含以下功能：

- 定义一个函数 `operateString`，该函数接受两个参数：一个字符串 `str` 和一个函数指针，函数指针指向一个字符串操作函数。该函数将函数指针所指向的操作函数应用于输入字符串，并返回操作后的结果。

- 定义两个字符串操作函数：

1. `reverseString`：将输入字符串进行逆序操作。

2. `countVowels`：统计输入字符串中元音字母的个数。

在 `main` 函数中，用户输入一个字符串，然后选择要执行的操作（逆序或统计元音字母个数）。根据用户的选择，使用 `operateString` 函数来应用相应的字符串操作函数，并打印结果。

三．理解下面程序，为程序写上注释，用文字描述该程序的执行过程以及执行结果

#include <stdio.h>

#include <stdlib.h>

struct ListNode {

int val;

struct ListNode\* next;

};

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

if (l1 == NULL) {

return l2;

}

if (l2 == NULL) {

return l1;

}

if (l1->val < l2->val) {

l1->next = merge(l1->next, l2);

return l1;

}

else {

l2->next = merge(l1, l2->next);

return l2;

}

}

struct ListNode\* mergeSort(struct ListNode\* head) {

if (head == NULL || head->next == NULL) {

return head;

}

struct ListNode\* slow = head;

struct ListNode\* fast = head->next;

while (fast != NULL && fast->next != NULL) {

slow = slow->next;

fast = fast->next->next;

}

struct ListNode\* right = slow->next;

slow->next = NULL;

struct ListNode\* left = mergeSort(head);

right = mergeSort(right);

return merge(left, right);

}

struct ListNode\* createNode(int val) {

struct ListNode\* newNode = (struct ListNode\*)malloc(sizeof(struct ListNode));

newNode->val = val;

newNode->next = NULL;

return newNode;

}

void printList(struct ListNode\* head) {

struct ListNode\* cur = head;

while (cur != NULL) {

printf("%d ", cur->val);

cur = cur->next;

}

printf("\n");

}

void freeList(struct ListNode\* head) {

struct ListNode\* cur = head;

while (cur != NULL) {

struct ListNode\* temp = cur;

cur = cur->next;

free(temp);

}

}

int main() {

struct ListNode\* head = createNode(4);

head->next = createNode(2);

head->next->next = createNode(1);

head->next->next->next = createNode(3);

printf("原始链表：");

printList(head);

head = mergeSort(head);

printf("排序后的链表：");

printList(head);

freeList(head);

return 0;

}

四、

1、说明define constant volatile这三个是干什么的

2、说出编译的五个主要过程

完成任务后将文件夹压缩成zip压缩包交至邮箱lochaoxive@163.com，压缩包命名规则为名字+年级+专业+班级+第几次任务（如：张三23电信1第一次任务）

截止时间：2023年11月24日16:59