吉林大学 2004C 语言程序设计试题 （含答案）

注意： 1.答案一律书写在答题纸上。
2.题签随答题纸交回。
3.对每道题都要写出其中重要变量的功能。
4.书写要保持整齐，否则会影响分数。
****

1.（30 分）我们称用 1 和 0 组成的串为“零幺串”，称只用 1 组成的串为“幺串”，称只用 0 组成的串为“零串”。试写一个函数（过程），使得它对任给零幺串 S,将返回一个值 N1 和值 N0，其中 N1 表示 S 中最长幺串的长度，N0 表示 S 中最长零串的长度。例如，假设 S 是下面表示的零幺串，则在 N1 中返回 4，在 N0 中返回 3
S=000101110011100011111
[参考答案]

```
/*==========================================================================*/
/*程序名称：2004_1.c */
/*程序目的：计算一个“零幺串”当中最长的零/幺串     */
/*Writen by Apechn ,Soft Lab of JLU */
/*==========================================================================*/
#include
#define MAX 100

void main(void)
{
char s[MAX]; /*  用来保存用户输入的"零/幺串" */
int i = 0; /*  计数器  */
int n0 = 0; /*  保存最长"零串"数 0 的个数  */
int n1 = 0; /*  保存最长"幺串"中 1 的个数  */
int temp0 = 0,temp1 = 0;

printf("请输入一个仅由 0/1 组成的字符串:\n");
scanf("%s",s); /*  由用户输入"零幺串" */

while (s)
{
if (s == '0')
temp0++;
if (temp0 > n0) /* n0 取已经扫描过的"零串"中 0 个数最多者  */
{
n0 = temp0;
temp1 = 0; /*  处理"零串"时将"幺串"计数的临时变量置零  */
}
```

```
if (s == '1')
temp1++;
if (temp1 > n1) /* n1 取已经扫描过的"零串"中 1 个数最多者 */
{
n1 = temp1;
temp0 = 0; /* 处理"幺串"时将"零串"计数的临时变量置零 */
}

i++;
}

printf("最长的\"零串\"中零的个数即 n0 = %d .\n",n0); /* 输出结果 */
printf("最长的\"幺串\"中幺的个数即 n1 = %d .\n",n1);
}
```

2.（30 分）多项式用链表示，例如“4X5+2X2+5X+6”被表示成下面形式：

试写一个函数，使得它对任给的两个多项式链表，形成多项式相加的链表，并返回指向该链表的指针值。例如，假设给定下面 P 和 Q 两个多项式链表，则产生 R 多项式链表，并返回 R 的指针值。注意：不能有零系数项。

[参考答案]

```
/*==========================================================================
=======*/
/*程序名称：2004_2.c                                                    */
/*程序目的：两个多项式相加                                              */
/*Writen by Apechn ,Soft Lab of JLU                                      */
/*==========================================================================
=======*/
#include
#include

struct list /* 节点结构声明 */
{
int power; /* 多项式中变量的幂 */
int coeff; /* 多项式中变量的系数 */
struct list *next;
};
typedef struct list node;
typedef node *poly;

/*--------------------------------------*/
/* 打印出一个多项式的各项                */
```

```
/*--------------------------------------*/
void printPoly(poly head)
{
poly pointer; /* 临时指针变量 */
pointer = head;

while (pointer != NULL) /* 打印各节点 */
{
printf("[%d,%d] ",pointer->coeff,pointer->power);

pointer = pointer->next;
}

printf("\n");
}

/*--------------------------------------------------------------------------*/
/* 从键盘输入一个多项式的项数和各项的幂和系数,多项式的系数不能为零 */
/*--------------------------------------------------------------------------*/
poly createPoly(poly head) /* 从键盘输入一个多项式的项数和各项的幂和系数 */
{
poly newNode; /* 临时节点 */
poly pointer;
int n,i;
int coeffTemp,powerTemp; /* 临时变量 */

head = (poly) malloc(sizeof(node)); /* 内存配置 */
if (head == NULL)
printf("Memory allocate Failure!!\n"); /* 内存配置失败 */
else
{
printf("请输入要创建的多项式的项数：\n");
scanf("%d",&n);

if (n == 0)
return NULL;
printf("请按 x 降幂依次输入各项的系数和幂：\n");

scanf("%d%d",&coeffTemp,&powerTemp);/* 输入多项式一项的系数和幂 */

head->coeff = coeffTemp; /* 定义首节点 */
head->power = powerTemp;
head->next = NULL;
```

```
pointer = head;
for (i = 1;i < n;i++) /* 依次定义其它节点 */
{
scanf("%d%d",&coeffTemp,&powerTemp); /* 输入各项的系数和幂 */
newNode = (poly) malloc(sizeof(node));

newNode->coeff = coeffTemp;
newNode->power = powerTemp;
newNode->next = NULL;

pointer->next = newNode; /* 将新定义的节点连接到原链表的表尾 */
pointer = newNode; /* pointer 指针后移 */
}
}

printPoly(head);

return head;
}


/*----------------------------------*/
/* 释放一个链表的空间 */
/*----------------------------------*/
void freeList(poly head)
{
poly pointer; /* 临时指针变量 */

while (head != NULL)
{
pointer = head;
head = head->next;
free(pointer);
}
}

/*-----------------------------------------*/
/* 计算两个多项式的和，并将结果存入另一个链表 */
/*-----------------------------------------*/
poly addPoly(poly head1,poly head2)
{
poly pointer1,pointer2,pointer,newNode,head;/* 临时指针变量 */
```

```
pointer1 = head1;
pointer2 = head2;
head = (poly) malloc(sizeof(node));
pointer = head;

while ((pointer1 != NULL) || (pointer2 != NULL)) /* 若两个多项式至少有一个没有访问结束 */
{
if (pointer1 !=NULL && pointer2 != NULL &&(pointer1->power > pointer2->power))
{/* 若两个多项式当前指针指向的项的幂不相等 */
newNode = (poly) malloc(sizeof(node));
newNode->power = pointer1->power;
newNode->coeff = pointer1->coeff;
newNode->next = NULL;

pointer->next = newNode;
pointer = newNode;

pointer1 = pointer1->next;
}

if (pointer1 !=NULL && pointer2 != NULL &&(pointer1->power < pointer2->power))
{/* 若两个多项式当前指针指向的项的幂不相等 */
newNode = (poly) malloc(sizeof(node));
newNode->power = pointer2->power;
newNode->coeff = pointer2->coeff;
newNode->next = NULL;

pointer->next = newNode;
pointer = newNode;/* 指向存储结果的链表的指针后移 */
pointer2 = pointer2->next;
}

if (pointer1 !=NULL && pointer2 != NULL &&(pointer1->power == pointer2->power))
{ /* 若两个多项式当前指针指向的项的幂相等 */
newNode = (poly) malloc(sizeof(node));
newNode->power = pointer1->power;
newNode->coeff = pointer1->coeff + pointer2->coeff;
newNode->next = NULL;

pointer->next = newNode;
pointer = newNode;/* 指向存储结果的链表的指针后移 */

pointer1 = pointer1->next;
pointer2 = pointer2->next;
```

```
}

if (pointer1 == NULL && pointer2 != NULL)
{
newNode = (poly) malloc(sizeof(node));
newNode->power = pointer2->power;
newNode->coeff = pointer2->coeff;
newNode->next = NULL;

pointer->next = newNode;
pointer = newNode;/*  指向存储结果的链表的指针后移  */

pointer2 = pointer2->next;
}

if (pointer2 == NULL && pointer1 != NULL)
{
newNode = (poly) malloc(sizeof(node));
newNode->power = pointer1->power;
newNode->coeff = pointer1->coeff;
newNode->next = NULL;

pointer->next = newNode;
pointer = newNode;

pointer1 = pointer1->next;
}

}


pointer = head; /*  由于多用了一个节点，这 3 行代码将多出的节点释放  */
head = head->next;
free(pointer);

return (head);
}

/*--------------------------------------------------------------------------*/
/*  主程序                                                                  */
/*--------------------------------------------------------------------------*/
void main()
{
poly p,q,r; /*  定义三个代表多项式的链表  */
```

```
printf("这是多项式 p:\n"); /*  创建多项式 p */
p = createPoly(p);

printf("这是多项式 q:\n"); /*  创建多项式 q */
q = createPoly(q);

r = (poly) malloc(sizeof(node));
printf("这是多项式 p 与多项式 q 的和 r:\n");
r = addPoly(p,q); /*  将多项式 p 和多项式 q 的和赋值给 r */
printPoly(r);

freeList(p); /*  释放程序中使用过的链表所占用的空间  */
freeList(q);
freeList(r);
}
```

3.（30 分）试写一个函数 f，使得它计算 sin(x)的近似值。其计算过程是 i 从 1 开始计算 Si(x)，直至|Si(x)-Si-1(x)|<0.00005,并把 Si 的值作为 sin(x)的近似值返回。其中
Si(x)=x-(x3/3!)+(x5/5!)-……+(-1)i-1(x2i-1/(2i-1)!)
不允许用库函数（abs 例外），而且要求写成递归函数(f)。
[参考答案]

```
/*==========================================================================
*/
/*程序名称：2004_3.c */
/*程序目的：求 sin(x)的近似值  */
/*Writen by Apechn ,Soft Lab of JLU */
/*==========================================================================
*/
#include
#include
#define MIN 0.00005 /*  设置运算停止条件  */

/*--------------------------------------------------------------------------*/
/*  计算弧度数为 angle 的角的正弦值第 i 项  */
/*--------------------------------------------------------------------------*/
float spill
```

4.（30 分）试写出函数 H（要求是递归函数），它对于任给的一棵树，同时求出其高度、结点个数和路径个数。其调用方式是这样：H(T,h,n,k),其中 T 是指向树根的指针变量，h 是返回树高度的整型变量，n 是返回结点个数的整型变量，k 是反贿赂经个数的整型变量。假设树结点具有下面几种结构之一（其中 son,son1 和 son2 表示儿子结点指针）：

在这里，数的高度是指最高层数减一，而路径是指从树根到某一叶结点的结点序列。例如，假设 T 指向下面图所示的树，则在执行 H(T,h,n,k)后返回时，将有 h=3,n=7,k=3。

[参考答案]

```
/*==========================================================================
*/
/*函数名称：2004_4.c */
/*函数目的：递归的求出一个给定结构树的高度、节点数和路径个数                    */
/*Writen by Siyee ,Soft Lab of JLU */
/*==========================================================================
*/
int depth = 0; /* 用来保存深度的变量 */

void H(tree *T, int *h, int *n, int *k)
{
++*n;
++depth;
*h = max(*h, depth); /* 如果 h 比 depth 小，则更新 h */
if(T->kind=='two'){ /* 若节点有两个子节点 */
H(T->son1, h, n, k); /* 递归计算两个子节点 */
H(T->son2, h, n, k);
}
else if(T->kind=='one'){ /* 若节点有一个子节点 */
H(T->son, h, n, k); /* 递归计算这一个子节点 */
}
else{
++*k; /* 如果是叶节点，则路径数 k 增一 */
}
--depth; /* 回溯，深度减一 */
}
```

5.（30 分）试写一个递归函数 F(递归过程），使得它对任给的整数链表，删除重复结点（值相同结点），并返回新链表的头地址。例如，假设有给定的整数链表：

则删除重复结点后得到如下链表：

注意，本题的要求是写出递归函数或递归过程。

[参考答案]

```
/*==========================================================================
*/
/*程序名称：2004_5.c */
/*程序目的：删除整数单链表中重复的节点          */
/*Writen by Apechn ,Soft Lab of JLU */
/*==========================================================================
*/
#include
```

```
#include
#define N 6 /*  节点个数  */

struct list /*  节点结构声明  */
{
int number;
struct list *next;
};
typedef struct list node;
typedef node *link;

int data[N] = {0,3,0,3,1,3}; /*  输入数据  */

/*--------------------------------------------------------------------------*/
/*  以已有的数组
```