题型一：分析算法功能

1. 针对如下算法，回答问题：

若数组A[n]={12,24,0,38,0,0,0,0,29,0,45,0}, n=12，给出算法执行后数组A[n]的状态。

template <class T>

void unknown(int A[], int n) {

int free=0;

for(int i=0; i<n; i++ )

if(A[i]!=0) {

if(i!=free) {

A[free]=A[i];

A[i]=0;

}

free++;

}

}

2. 针对如下算法，回答问题：

(1) 若整型数组A[8]={12,24,33,38,95,83,64,57}，n=8，则给出算法返回的结果。

(2) 说明算法的功能是什么。

int unknown(int A[], int n) {

if(n==1) return A[0];

int temp=unknown(A, n-1);

return A[n-1]>temp? A[n-1] : temp;

}

3. 读下面的程序段，说明程序所完成的功能。

int A(LinkList &L) { /\*L是无表头结点的单链表\*/

if ( L&&L->next){

Q=L; L=L->next; P=L;

while (P->next) P=P->next;

P->next=Q; Q->next=NULL;

}

return 1;

}

4.给出下列递归过程的执行结果。

(1) void *unknown* ( int *w* ) {

if ( *w* ) {

*unknown* ( *w*-1 );

for ( int *i* = 1; i <= *w*; *i*++ ) cout << *w*;

cout << endl;

}

}

调用语句为 *unknown* (4)。

(2) void *unknown* ( int *m* ) {

cout <<  *n* % 10 ;

if ( int ( *n* / 10 ) ) *unknown* ( int ( *n* / 10 ) );

}

调用语句为*unknown* ( 582 )。

(3) int *unknown* ( int *m* ) {

int *value*;

if ( ! *m* ) *value* = 3;

else *value* = *unknown* ( *m*-1 ) + 5;

return *value*;

}

执行语句为 cout << *unknown* (3)。

5.阅读下面算法，按要求作答，其中Push(S, d)表示将数据元素d压入栈S中，Pop(T,d)表示T的栈顶元素出栈并存入d中。

int algo (Stack S , int e)

{

int d; Stack T;

InitStack(T);

while(!StackEmpty(S)) {

Pop(S,d);

if(d!=e) Push(T, d);

} //while

while(!StackEmpty(T)) {

Pop(T, d);

Push(S, d);

}

}

（1）假设栈S中的原始数据从栈底至栈顶依次为：3，5，7，12，5，6，8；e的值为5。请写出算法执行完后栈S中从栈底至栈顶的数据元素序列。

（2）简述该算法的功能。

6.已知数组a中存放着两组数据元素序列（a0，a1，．．．，am-1，b0，b1，．．．，bn-1）。下面算法利用原存储空间将a中的数据元素序列就地互换为（b0，b1，．．．，bn-1，a0，a1，．．．，am-1），算法思想可以描述为：

（1）把数组a中的数据元素序列（a0，a1，．．．，am-1，b0，b1，．．．，bn-1）就地逆置为（bn-1，．．．，b1，b0，am-1，．．．，a1，a0）。

（2）把数组a中的数据元素序列（bn-1，．．．，b1，b0，am-1，．．．，a1，a0）就地逆置为（b0，b1，．．．， bn-1，am-1，．．．，a1，a0）。

(3) 把数组a中的数据元素序列（b0，b1，．．．， bn-1，am-1，．．．，a1，a0）就地逆置为（b0，b1，...， bn-1，a0，a1...，am-1）。

根据上述算法思想，请在空缺处填上适当的语句，以完善算法功能。

void converse (ElemType a[],int low,int high)

{ //将数组a中自下标low 至high的一段数据元素逆置

int n,i;

ElemType x;

n= (high-low+1)/2; // n 为循环次数

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

{ x=a[low+i];

①\_\_\_\_\_\_\_\_\_\_\_\_;

②\_\_\_\_\_\_\_\_\_\_\_\_;

}

}

void exchange (ElemType a[],int m,int n)

{

converse(a,0,m+n-1);//将数组a中的m+n个元素就地逆置

③\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

④\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

}

7.设单链表结构为 struct *ListNode* {

int *data* ;

*ListNode* \* *link* ;

};

下面的程序是以单链表为存储结构, 实现二路归并排序的算法, 要求链表不另外占用存储空间, 排序过程中不移动结点中的元素, 只改各链结点中的指针, 排序后r仍指示结果链表的第一个结点。在初始状态下, 所有待排序记录链接在一个以r为头指针的单链表中。例如,



在算法实现时，利用了一个队列做为辅助存储, 存储各有序链表构成的归并段的链头指针。初始时, 各初始归并段为只有一个结点的有序链表。队列的数据类型为*Queue*, 其可直接使用的相关操作有

◼ 置空队列操作：*makeEmpty* ( );

◼ 将指针*x*加入到队列的队尾操作：*EnQueue* ( *ListNode* \* *x* );

◼ 退出队头元素, 其值由函数返回的操作：*ListNode* \**DlQueue* ( );

◼ 判队列空否的函数, 空则返回1, 不空则返回0：int *IsEmpty*( )。

解决方法提示：

⮚ 程序首先对待排序的单链表进行一次扫描, 将它划分为若干有序的子链表, 其表头 指针存放在一个指针队列中。

⮚ 当队列不空时, 从队列中退出两个有序子链表, 对它们进行二路归并, 结果链表的表头指针存放到队列中。

⮚ 如果队列中退出一个有序子链表后变成空队列, 则算法结束。这个有序子链表即为所求。

在算法实现时有 6 处语句缺失，请阅读程序后补上。

(1) 两路归并算法

void *merge* ( *ListNode \* ha*, \* *hb*;  *ListNode \**& *hc* ) {

*ListNode \*pa, \*pb, \*pc* ;

if ( *ha*→*data* <= *hb*→*data* )

{ *hc* = *ha*; *pa* = *ha*→*link*; *pb* = *hb*; }

else { *hc* = *hb*; *pb* = *hb*→*link*; *pa* = *ha* ; }

*pc* = *hc*;

while ( *pa* && *pb* )

if ( (1) ) {

*pc*→*link* = *pa*; *pc* = *pa*; (2) ;

}

else {

*pc*→*link* = *pb*; *pc* = *pb*; (3) ;

}

if ( *pa* ) *pc*→*link* = *pa*;

else *pc*→*link* = *pb*;

};

(2) 归并排序主程序

void *mergesort* ( *ListNode* \* *r* ) {

*ListNode* \* *s*, *t*; *Queue* *Q* ;

if ( ! *r* ) return;

*s* = *r*; (4) ;

while ( *s* ) {

*t* = *s*→*link*;

while ( *t* != 0 && *s*→*data* <= *t*→*data* ) { *s* = *t*; *t* = *t*→*link*; }

if ( *t* ) {

*s*→*link* = 0; *s* = *t*; (5) ;

}

}

while ( !*Q.IsEmpty*( ) ) {

*r* = *Q*.*DlQueue*( );

if ( *Q.IsEmpty*( ) ) break;

*s* = *Q*.*DlQueue*( );

*merge*( *r*, *s*, *t* ); (6) ;

}

}

题型二 简答题

(1) 在一个有*n*个元素的顺序表的第*i*个元素（1 ≤ *i* ≤ *n*）之前插入一个新元素时，需要向后移动多少个元素？

(2) 当一个栈的进栈序列为1234567时，可能的出栈序列有多少种？6457321是否是合理的出栈序列？

题型三 程序设计题

1. 试设计算法，将对带头结点的单链表实现就地逆置。单链表及结点定义如下：

typedef struct LNode {

ElemType data；

struct LNode \*next；

}LNode，\*LinkList；

2. 假设以带头结点的循环链表表示队列，并且只设一个指针指向队尾元素结点（注意不设头指针），试编写相应的队列初始化、入队列和出队列的算法。

/\*------队列的循环链表存储结构----\*/

typedef struct node {

ElemType data;

struct node \*next;

}Node,\*Link; //结点类型

typedef struct {

Link rear;

}Que; //队列类型

/\*-----队列初始化-----\*/

Status Init(Que &q)

{

}

/\*-------入队列---------\*/

Status EnQue(Que &q,ElemType x)

{

}

/\*-------出队列------\*/

Status DeQue(Que &q, ElemType &e)

{

}