一、选择题

1、从逻辑上可以把数据结构分为（ ）两大类。

A．线性结构、非线性结构 B．顺序结构、链式结构

C．动态结构、静态结构 D．初等结构、构造型结构

2、下面关于线性表的叙述中，错误的是哪一个？（ ）

A．线性表采用顺序存储，必须占用一片连续的存储单元。

B．线性表采用链接存储，便于插入和删除操作。

C．线性表采用链接存储，不必占用一片连续的存储单元。

D．线性表采用顺序存储，便于进行插入和删除操作。

3、若某线性表最常用的操作是存取任一指定序号的元素和在最后进行插入和删除运算，则利用（ ）存储方式最节省时间。

A．带头结点的双循环链表 B．双链表 C．顺序表 D．单循环链表

4、某线性表中最常用的操作是在最后一个元素之后插入一个元素和删除第一个元素，则采用（ ）存储方式最节省运算时间。

A．单链表 B．仅有尾指针的单循环链表

C．双链表 D． 仅有头指针的单循环链表

5、链表不具有的特点是（ ）

A．可随机访问任一元素 B．插入、删除不需要移动元素

C．不必事先估计存储空间 D．所需空间与线性长度成正比

6、以下与数据的存储结构无关的术语是（ ）。

A．栈 B. 链表 C. 顺序表 D. 循环队列

7、在一个单链表中，若删除\*p结点的后面一个结点（存在），则执行的语句是（ ）

A p->next=p->next－>next; free(p->next);

B p->next=s; s->next=p->next; free(s);

C p->next=s->next; s=p->next;free(s);

D s=p->next;p->next=s->next; free(s)；

8、一个栈的入栈序列是a,b,c,d,e,f，则栈的可能的输出序列是（ ）：

A fcdabe B dfcabe C abcdef D fdbace

9、以下哪个是循环队列Q的队满条件（ ）：

A （Q.rear+1）% maxsize = = (Q.front + 1 ) % maxsize；

B （Q.rear+1）% maxsize = = Q.front;

C Q.rear = = Q.front;

D （Q. front +1）% maxsize = = Q. rear ;

10、循环队列A[0..m-1]存放其元素值，用front和rear分别表示队头和队尾，则当前队列中的元素数是( )。

A. (rear-front+m)%m B. rear-front+1

C. rear-front-1 D. rear-front

11、下面程序段中state语句的执行次数为（ ）：

for(i=1;i<＝n＋1;i++)

{j=0;

while(j<＝i)

{state; j++;}}

A （n＋1）(n+4)/2 B (n＋1)(n+2)/2 C (n＋1)(n+4) D n(n+3)/2

12、若长度为n的线性表采用顺序存储结构，在其第i个位置插入一个新元素的算法的时间复杂度为（ ）(1<=i<=n+1)。

A. O(0) B. O(1) C. O(n2) D. O(n)

13、一个栈的输入序列为1 2 3 4 5，则下列序列中不可能是栈的输出序列的是（ ）。

A. 2 3 4 1 5 B. 2 3 1 4 5 C. 5 4 1 3 2 D. 1 5 4 3 2

14. 输入序列为ABC，可以变为BCA时，经过的栈操作为（ ）。

A. push,pop,push,pop,push,pop B. push,push,pop,push,pop,pop

C. push,push,push,pop,pop,pop D. push,pop,push,push,pop,pop

15. 设计一个判别表达式中左，右括号是否配对出现的算法，采用（ ）数据结构最佳。

A．栈 B. 队列 C. 线性表的链式存储结构 D. 线性表的顺序存储结构

答案：

1-5：ADCBA 6-10：ADCBA 11-15：ADCBA

二、填空题

1.当线性表的元素总数基本稳定，且很少进行插入和删除操作，但要求以最快的速度存取线性表中的元素时，应采用 存储结构。

2.对一个算法，随着其处理问题的数据增大，处理会越来越困难复杂，人们通常把描述数据增大程度的量叫做 。

3．在一个长度为n的顺序表中第i个元素（1<=i<=n）之前插入一个元素时，需向后移动\_\_\_\_\_ \_\_\_个元素。

4．对于双向链表,在两个结点之间插入一个新结点需修改的指针共 个，单链表为 个。

5．设单链表的结点结构为(data,next)，next为指针域，已知指针p指向单链表中data为x的结点，指针s指向data为y的新结点 , 若将结点s插入结点p之后，则需要执行以下语句:\_\_\_\_ \_\_\_； \_\_\_ \_\_ \_;

6. 又称作先进先出表。

7．在单链表L中，指针p所指结点有后继结点的条件是：

8.带头结点的单循环链表L为空表的条件是： 。

9．当两个栈共享一存储区时，栈利用一维数组stack[M]表示，两栈顶指针为top1与top2，则当栈1空时，top1为 ，栈2空时 ，top2为 ,栈满的条件是： 。

答案：

1. 顺序 2. 问题规模 3. n-i+1 4. 4个，2个 5. s->next=p->next; p->next=s

6. 队列 7. p->next!=NULL 8. L->next==L; 9. -1; M; top2==top1+1

三、判断题

1. 对任何数据结构链式存储结构一定优于顺序存储结构。( )

2. 顺序存储结构的主要缺点是不利于插入或删除操作。( )

3.两个栈共享一片连续内存空间时，为提高内存利用率，减少溢出机会，应把两个栈的栈底分别设在这片内存空间的两端。（ ）

4．健壮的算法不会因非法的输入数据而出现莫名其妙的状态。( )

5．算法的优劣与算法描述语言无关，但与所用计算机有关。( )

6．线性表的逻辑顺序总是与其物理顺序一致。（ ）

7．程序一定是算法。( )

8．数据的物理结构是指数据在计算机内的实际存储形式。( )

9. 顺序存储方式的优点是存储密度大，且插入、删除运算效率高。( )

10. 栈和队列的存储方式，既可以是顺序方式，又可以是链式方式。（ ）

11. 线性表只能用顺序存储结构实现。( )

12. 栈和队列都是线性表，只是在插入和删除时受到了一些限制。（ ）

13. 数据的逻辑结构说明数据元素之间的顺序关系,它依赖于计算机的储存结构. ( )

14. 链表是采用链式存储结构的线性表,进行插入、删除操作时，一般情况下在链表中比在顺序存储结构中效率高。 ( )

15. 线性表的特点是每个元素都有一个前驱和一个后继。( )

答案：

1-5：F T T T F 6-10: F F T F T 11-15: F T F T F

四、解答题

1、写出双向链表中结点的定义，设结点含有3个域：一个整型域data，两个指针域prior和next。

答案：

typedef int ElemType;

typedef struct DLNode

{ElemType data;

struct DLNode \*prior,\*next;

}DLNode,\*DLinkList;

2、画图表示，并写出在上述双向链表中一已知结点p后插入结点s的四条语句，请标明先后顺序。

S->next=p->next;

S->prior=p;

p->next->prior=s;

P->next=s;

3、画图表示，并写出在上述双向链表中删除已知结点q（q结点存在）的语句

五、算法设计题

1. 已知一带头结点的双向循环链表L，结点类型同第四题中的说明（不需重新定义），编写函数判断该链表中的元素是否对称，对称返回1，否则返回0。

typedef int ElemType;

typedef struct DLNode

{ElemType data;

struct DLNode \*prior,\*next;

}DLNode,\*DLinkList;

int duichen(DLinkList L)

{

DLinkList p, q;

p=L->next;

q=L->prior;

while(p!=q)

{

if(p->data!=q->data)

return 0;

else

if(p->next==q) return 1;

else

{

p=p->next;

q=q->prior;

}

}

return 1;

}

2、带头结点的单链表L中的结点值域按非递减排列，请编写算法，使得L变为递增的单链表（假设原来单链表的值为1，3，3，4，5，6，6，处理后为1，3，4，5，6）。结点类型以讲义为主，不需要重新定义。

typedef int ElemType;

typedef struct node {

ElemType data;

struct node \*next;

}LNode, \*LinkList；

void deletechongfu(LinkList L)

{

LinkList p;

if(L->next==NULL||L->next->next==NULL) //只有0个或1个元素，直接返回

return;

pre=L->next;

p=pre->next;

while(p!=NULL)

{

if(p->data==pre->data) //相等则删除

{

pre->next=p->next;

free(p);

p=pre->next;

}

else

{

pre=p;

p=p->next;

}

}

return;

}