数据结构试题及答案

一、单项选择题

(1)          一个算法应该是（ ）。

A) 程序                                                      B) 问题求解步骤的描述

C) 要满足五个基本属性                         D) A和C

(2)          算法指的是（  ）。

A) 计算机程序                                            B) 解决问题的计算方法

C) 排序算法                                               D) 解决问题的有限运算序列。

(3)   与数据元素本身的形式、内容、相对位置、个数无关的是数据的（ ）。

A) 存储结构            B) 逻辑结构            C) 算法                   D)操作

(4)   从逻辑上可以把数据结构分为（ ）两大类。

A) 动态结构、静态结构                            B) 顺序结构、链式结构

C) 线性结构、非线性结构                          D) 初等结构、构造型结构

(5)  下列叙述中正确的是( )。

　　A）一个逻辑数据结构只能有一种存储结构

　　B）数据的逻辑结构属于线性结构，存储结构属于非线性结构

　　C）一个逻辑数据结构可以有多种存储结构，且各种存储结构不影响数据处理的效率

D）一个逻辑数据结构可以有多种存储结构，且各种存储结构影响数据处理的效率

(6)  数据的基本单位是（　　）

       A) 数据项 B) 数据类型 C) 数据元素 D) 数据变量

(7)  设有一个递归算法如下：

int fact(int n)

{  /\*  大于等于0   \*/

    if ( n<=0 )  return  1  ;

    else  return  n\*fact (n-1)  ;

}

则计算fact(n)需要调用该函数的次数为（  B   ）。

1. n                     B) n+1                   C) n+2                 D) n-1

(8) 若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，则最节省运算时间的存储方式是（　 ）。

A)单链表                                                       B)仅有头指针的单循环链表

C)双链表                                                        D)仅有尾指针的单循环链表

(9) 非空的单循环链表的头指针为head,尾指针为rear，则下列条件成立的是（ ）。

     A) rear->next= =head                                         B) rear->next->next= =head

     C) head->next= =rear                                         D) head->next->next= =rear

(10)从一个长度为n的顺序表中删除第i个元素（1≤i≤n）时，需向前移动的元素的个数是（ ）。

A)n-i  B)n-i+1  C)n-i-1    D)i

(11)  已知一个有序表为（13，18，24，35，47，50，62，83，90，115，134），当二分检索值为90的元素时，检索成功需比较的次数是（　　）。

A)1 B)2  C)3  D)4

(12) 假设以行优先顺序存储三维数组R[6][9][6]，其中元素R[0][0][0]的地址为2100，且每个元素占4个存储单元，则存储地址为2836的元素是（　　）。

A) R[3][3][3]   B) R[3][3][4] C) R[4][3][5]  D) R[4][3][4]

(13)       线性表采用链式存储时，节点的存储的地址（   ）。

A) 必须是不连续的      B) 连续与否均可

C) 必须是连续的         D) 和头节点的存储地址相连续

(14)       用链表表示线性表的优点是（  ）。

A) 便于随机存取                               B) 花费的存储空间比顺序表少

C) 数据元素的物理顺序与逻辑顺序相同     D) 便于插入与删除

(15)       链表不具有的特点是（  ） 。

A) 插入、删除不需要移动元素    B) 可随机访问任一元素

C) 不必事先估计存储空间  D) 所需空间与线性长度成正比

(16) 在长度为n的顺序表中删除第i个元素(1≤i≤n)时，元素移动的次数为( )。

A) n-i+1 B) i C) i+1 D) n-i

(17) 采用顺序搜索方法查找长度为n的顺序表示，搜索成功的平均搜索长度为（  ）。

A) n B) n/2 C) (n-1)/2 D) (n+1)/2

(18)  若不带头结点的单链表的头指针为head，则该链表为空的判定条件是( )。

A) head==NULL B) head->next==NULL

C) head!=NULL　　     D) head->next==head

(19) 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) 仅有头指针的单循环链表

C) 双链表                                                   D) 仅有尾指针的单循环链表

(20) 若允许表达式内多种括号混合嵌套，则为检查表达式中括号是否正确配对的算法，通常选用的辅助结构是（　　）。

       A) 栈 B) 线性表 C) 队列   D) 二叉排序树

(21)顺序栈S中top为栈顶指针，指向栈顶元素所在的位置，elem为存放栈的数组，则元素e进栈操作的主要语句为（　 　）。

A) s.elem［top］=e；   s.top=s.top+1；

B) s.elem［top+1］=e；s.top=s.top+1；

C) s.top=s.top+1； s.elem［top+1］=e；

D) s.top=s.top+1；s.elem［top］=e；

(22)  循环队列sq中，用数组elem［0··25］存放数据元素，sq.front指示队头元素的前一个位置，sq.rear指示队尾元素的当前位置，设当前sq.front为20，sq.rear为12，则当前队列中的元素个数为（　 　）。

A) 8                            B) 16                          C) 17                          D) 18

(23)   链式栈与顺序栈相比，一个比较明显的优点是（  ）。

A) 插入操作更加方便       B) 通常不会出现栈满的情况

C) 不会出现栈空的情况   D) 删除操作更加方便

(24) 一个递归的定义可以用递归过程求解，也可以用非递归过程求解，但单从运行时间来看，通常递归过程比非递归过程（ ）。

A) 较快  B) 较慢                C) 相同                 D) 不定

(25) 若已知一个栈的入栈序列是1,2,3,4……n，其输出序列为p1,p2,p3,……pn，若p1= =n,则pi为（ ）。

       A) i                        B) n= =i                C) n-i+1                 D) 不确定

(26) 一个栈的入栈序列是a,b,c,d,e，则栈的不可能的输出序列是( ) 。

A) edcba                   B) decba                   C) dceab                   D) abcde

(27)  若进栈序列为1，2，3，4，5，6，且进栈和出栈可以穿插进行，则不可能出现的出栈序列是(  )。

A) 2，4，3，1，5，6　　                              B) 3，2，4，1，6，5

C) 4，3，2，1，5，6 　　                             D) 2，3，5，1，6，4

(28) 对于栈操作数据的原则是（ ）。

A) 先进先出 B) 后进先出 C) 后进后出    D) 不分顺序

(29) 栈和队列的共同点是（  ）。

A) 都是先进先出                          B) 都是先进后出

C) 只允许在端点处插入和删除元素          D) 没有共同点

(30)  一个队列的入队序列是1，2，3，4，则队列的输出序列是（ ）。

A) 4,3,2,1                B) 1,2,3,4                C)1,4,3,2              D) 3,2,4,1

(31)   设数组data[m]作为循环队列SQ的存储空间，front为队头指针，rear为队尾指针，则执行出对操作后其头指针front值为（ ）。

A)  front=front+1             B)  front=(front+1)%(m-1)

C)  front=(front-1)%m     D)  front=(front+1)%m

(32)引起循环队列队头位置发生变化的操作是( )。

A) 出队 B) 入队 C) 取队头元素   D) 取队尾元素

(33)  设以数组A[m]存放循环队列的元素，其头尾指针分别为front和rear，则当前队列中的元素个数为（  ）。

A）(rear-front+m)%m  B）rear-front+1

C）(front-rear+m)%m  D）(rear-front)%m

(34)二维数组A[12][18]采用列优先的存储方法，若每个元素各占3个存储单元，且A[0][0]地址为150，则元素A[9][7]的地址为(   )。

A) 429　 B) 432　 C) 435 D) 438

(35)  树中所有结点的度之和等于所有结点数加（  ）。

A) 0                   B)1                       C) -1                      D) 2

(36)  在一棵具有n个结点的二叉链表中，所有结点的空域个数等于（   ）。

A) n                   B) n-1                   C) n+1                   D) 2\*n

(37) 某二叉树的先序序列和后序序列正好相反，则该二叉树一定是 （  ）的二叉树。

A) 空或只有一个结点                               B) 高度等于其节点数

C) 任一结点无左孩子                               D) 任一结点无右孩子

(38) 含有10个结点的二叉树中，度为0的结点数为4，则度为2的结点数为（　　　）

A)3                              B)4                              C)5                              D)6

(39) 除第一层外，满二叉树中每一层结点个数是上一层结点个数的（　　　）

A)1/2倍                       B)1倍                          C) 2倍                         D) 3倍

(40) 对一棵有100个结点的完全二叉树按层编号，则编号为49的结点，它的父结点的编号为（　　　）

A)24                            B)25                            C)98                            D)99

(41)       可以惟一地转化成一棵一般树的二叉树的特点是（　　　）

A)根结点无左孩子       B)根结点无右孩子

C)根结点有两个孩子    D)根结点没有孩子

(42) 设高度为h的二叉树上只有度为0和度为2的结点，则此类二叉树中所包含的结点数至少为（   ）。

A) 2h                      B) 2h-1                   C) 2h+1                  D) h+1

(43)       在一棵度为3的树中，度为3的节点个数为2，度为2的节点个数为1，则度为0的节点个数为（   ）。

       A) 4                        B) 5                        C) 6                        D) 7

(44)       将一株有100个节点的完全二叉树从上到下，从左到右依次进行编号，根节点的编号为1，则编号为49的节点的 左孩子编号为（ ）。

A) 98 B) 89 C) 50 D) 没有孩子

(45)       树最适合用来表示（  C ）。

A) 有序数据元素                            B) 无序数据元素

      C) 元素之间具有分支层次关系的数据     D) 元素之间无联系的数据

(46)       在一个非空二叉树的中序遍历序列中，根结点的右边（  ）。

       A) 只有右子树上的所有结点   B) 只有右子树上的部分结点

C) 只有左子树的上的部分结点 D) 只有左子树上的所有结点

(47) 任何一棵二叉树的叶结点在先序、中序和后序遍历序列中相对次序（  ）。

       A) 不发生改变 B) 发生改变 C) 不能确定   D) 以上都不对

(48) 在有n个叶子结点的哈夫曼树中，其结点总数为（ D  ）。

       A) 不确定               B) 2n                       C) 2n+1                  D) 2n-1

(49)    权值为{1,2,6,8}的四个结点构成的哈夫曼树的带权路径长度是（ ）。

       A) 18                       B) 28                       C) 19                       D) 29

(50)       对一个满二叉树，m个树叶，k个分枝结点，n个结点，则（  ）。

       A) n=m+1                B) m+1=2n              C) m=k-1                 D) n=2k+1

(51) 若采用邻接矩阵翻存储一个n个顶点的无向图，则该邻接矩阵是一个（ ）。

       A) 上三角矩阵       B) 稀疏矩阵           C) 对角矩阵        D) 对称矩阵

(52)       在一个图中，所有顶点的度数之和等于所有边数的（  ）倍。

       A) 1/2                      B) 1                        C) 2                        D) 4

(53)  在一个有向图中，所有顶点的入度之和等于所有顶点的出度之和的（  ）倍。

       A) 1/2                      B) 1                        C) 2                        D) 4

(54)    如果求一个连通图中以某个顶点为根的高度最小的生成树，应采用（　　　）。

A)   深度优先搜索算法                                   B)    广度优先搜索算法

C) 求最小生成树的prim算法                         D) 拓扑排序算法

(55) n个顶点的连通图至少中含有(  )条。

       A) n-1                     B) n                        C) n+1                    D) 0

(56) n个顶点的完全有向图中含有(  )。

A) n-1条有向边　　　B) n条有向边

C) n(n-1)/2条有向边　D) n(n-1)条有向边

(57) 在无向图中定义顶点Vi域Vj之间的路径为从Vi到达Vj的一个（  ）。

A) 顶点序列           B) 边序列             C) 权值总和        D) 边的条数

(58) 无向图G=(V,E),其中：V={a,b,c,d,e,f}, E={(a,b),(a,e),(a,c),(b,e),(c,f), (f,d),(e,d)}，对该图进行深度优先遍历，得到的顶点序列正确的是（ ）。

A) a,b,e,c,d,f B) a,c,f,e,b,d C) a,e,b,c,f,d  D) a,e,d,f,c,b

(59) 下面哪一方法可以判断出一个有向图是否有环（回路）。

A) 求节点的度         B) 拓扑排序

 C) 求最短路径     D) 求关键路径

1. 图的广度优先搜索类似于树的（ ）次序遍历。
2. 先根                 B) 中根               C) 后根                D) 层次

(61)已知有向图G=(V,E)，其中V={V1,V2,V3,V4,V5,V6,V7}，E={<V1,V2>,<V1,V3>,<V1,V4>,<V2,V5>,<V3,V5>,<V3,V6>,<V4,V6>,<V5,V7>,<V6,V7>},G的拓扑序列是（ ）。

A) V1,V3,V4,V6,V2,V5,V7                         B) V1,V3,V2,V6,V4,V5,V7

C) V1,V3,V4,V5,V2,V6,V7                         D) V1,V2,V5,V3,V4,V6,V7

(62) 关键路径是事件结点网络中（ ）。

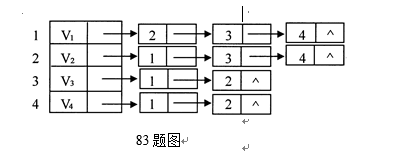
A) 从源点到汇点的最长路径 B) 从源点到汇点的最短路径

C) 最长回路                                  D) 最短回路

(63)有n个结点的有向完全图的弧数是（　 　）。

A) n2                            B) 2n                           C) n(n-1)                      D) 2n(n+1)

(64)设图的邻接链表如题12图所示，则该图的边的数目是（　　 　）。



1. 4                            B) 5                            C) 10                           D) 20

(65) 在一个图中，所有顶点的度数之和等于图的边数的（  ）倍。

A) 1/2                 B)  1                 C)  2                 D)  4

(66)在一个有向图中，所有顶点的入度之和等于所有顶点的出度之和的（  ）倍。

A) 1/2                 B)  1                 C)  2                 D)  4

(67) 有8个结点的无向图最多有（   ）条边。

A) 14                  B)  28               C)  56               D)  112

(68) 有8个结点的无向连通图最少有（    ）条边。

A) 5                    B)  6                 C)  7                 D)  8

(69)有8个结点的有向完全图有（  ）条边。

A) 14                  B)  28               C)  56               D)  112

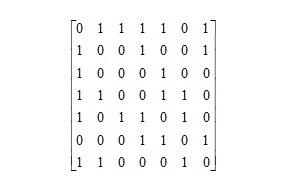
(70) 用邻接表表示图进行广度优先遍历时，通常是采用（   ）来实现算法的。

A) 栈                 B) 队列              C)  树               D)  图

(71) 用邻接表表示图进行深度优先遍历时，通常是采用（   ）来实现算法的。

A) 栈                 B) 队列              C)  树               D)  图

(72) 已知图的邻接矩阵，根据算法思想，则从顶点0出发按深度优先遍历的结点序列是（   ）。



1. 0 2 4 3 1 5 6             B)0 1 3 6 5 4 2

C)  0 4 2 3 1 6 5         D) 0 3 6 1 5 4 2

建议：0 1 3 4 2 5 6

(73)已知图的邻接矩阵同上题，根据算法，则从顶点0出发，按深度优先遍历的结点序列是（   ）。

A) 0 2 4 3 1 5 6        B)  0 1 3 5 6 4 2

C)  0 4 2 3 1 6 5        D)  0 1 3 4 2 5 6

(74已知图的邻接矩阵同上题，根据算法，则从顶点0出发，按广度优先遍历的结点序列是（    ）。

1. 0 2 4 3 6 5 1        B)  0 1 3 6 4 2 5

C)  0 4 2 3 1 5 6        D)  0 1 3 4 2 5 6

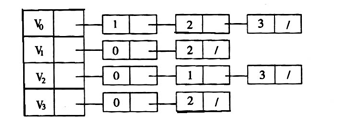
（建议：0 1 2 3 4 5 6）

(75) 已知图的邻接矩阵同上题，根据算法，则从顶点0出发，按广度优先遍历的结点序列是（    ）。

A) 0 2 4 3 1 6 5         B)  0 1 3 5 6 4 2

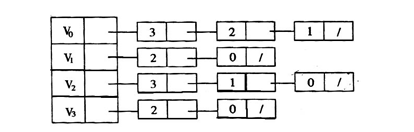
C)  0 1 2 3 4 6 5           D)  0 1 2 3 4 5 6

(76) 已知图的邻接表如下所示，根据算法，则从顶点0出发按深度优先遍历的结点序列是（   ）。



A)  1 3 2                B)  0 2 3 1               C)  0 3 2 1            D)  0 1 2 3

(79)   已知图的邻接表如下所示，根据算法，则从顶点0出发按广度优先遍历的结点序列是（   ）。



A) 0 3 2 1                B)  0 1 2 3               C)  0 1 3 2              D)  0 3 1 2

(80)       深度优先遍历类似于二叉树的（  ）。

A) 先序遍历 B)  中序遍历 C)  后序遍历 D)  层次遍历

(81)       广度优先遍历类似于二叉树的（    ）。

A) 先序遍历 B)  中序遍历 C)  后序遍历  D)  层次遍历

(82)       任何一个无向连通图的最小生成树（   ）。（注:生成树不唯一，但最小生成树唯一，即边权之和或树权最小的情况唯一）

A) 只有一棵 B)  一棵或多棵

C)  一定有多棵 D)  可能不存在

(83)   在分析折半查找的性能时常常加入失败节点，即外节点，从而形成扩充的二叉树。若设失败节点i所在层次为Li，那么查找失败到达失败点时所做的数据比较次数是（   ）。

A)  Li+1                 B)  Li+2                  C)  Li-1                  D)  Li

(84)   向一个有127个元素原顺序表中插入一个新元素并保存原来顺序不变，平均要移动（   ）个元素。

       A) 8                         B) 63.5                     C) 63                       D) 7

(85)   由同一组关键字集合构造的各棵二叉排序树(   )。

A) 其形态不一定相同，但平均查找长度相同

B) 其形态不一定相同，平均查找长度也不一定相同

C) 其形态均相同，但平均查找长度不一定相同

D) 其形态均相同，平均查找长度也都相同

(86)  能进行二分查找的线性表,必须以（ 　）。

A) 顺序方式存储，且元素按关键字有序

B) 链式方式存储，且元素按关键字有序

C) 顺序方式存储，且元素按关键字分块有序

D) 链式方式存储，且元素按关键字分块有序

(87)为使平均查找长度达到最小,当由关键字集合{05,11,21,25,37,40,41,62,84}构建二叉排序树时,第一个插入的关键字应为（ ）

A) 5 B)37 C) 41 D) 62

(88)对关键字序列(56,23,78,92,88,67,19,34)进行增量为3的一趟希尔排序的结果为　( 　)。

A) (19,23,56,34,78,67,88,92) 　                   B) 23,56,78,66,88,92,19,34)

C) (19,23,34,56,67,78,88,92)　                    D) (19,23,67,56,34,78,92,88)

(89)用某种排序方法对关键字序列{35,84,21,47,15,27,68,25,20}进行排序时，序列的变化情况如下：

20,15,21,25,47,27,68,35,84

15,20,21,25,35,27,47,68,84

15,20,21,25,27,35,47,68,84

则采用的方法是（   ）。

1. 直接选择排序 B) 希尔排序 C) 堆排序 D) 快速排序

(90)   一组记录的排序码为(46,79,56,38,40,84)，则利用快速排序的方法，以第一个记录为基准得到的第一次划分结果为（ ）。

       A) 38,40,46,56,79,84    B) 40,38,46,79,56,84

C) 40,38,46,56,79,84    D) 40,38,46,84,56,79

(91)  对待排序的元素序列进行划分，将其分为左、右两个子序列，再对两个子序列进行同样的排序操作，直到子序列为空或只剩下一个元素为止。这样的排序方法是（ ）。

A) 直接选择排序      B) 直接插入排序

C) 快速排序            D) 冒泡排序

(92)  将5个不同的数据进行排序，至多需要比较（  ）次。

A)  8               B)  9                C)  10                  D)  25

(93) 排序算法中，第一趟排序后，任一元素都不能确定其最终位置的算法是（　 　　）。

A)选择排序                 B)快速排序

C)冒泡排序                  D)插入排序

(94)   排序算法中，不稳定的排序是（　 　　）。

A)直接插入排序          B)冒泡排序

C)堆排序                     D)选择排序

(95)   排序方法中，从未排序序列中依次取出元素与已排序序列（初始时为空）中的元素进行比较，将其放入已排序序列的正确位置上的方法，称为（  ）.

A)  希尔排序         B)  冒泡排序        C)  插入排序       D)  选择排序

(96)  从未排序序列中挑选元素，并将其依次插入已排序序列（初始时为空）的一端的方法，称为（  ）。

A)  希尔排序         B)  归并排序        C)  插入排序       D)  选择排序

(97)   对ｎ个不同的排序码进行冒泡排序，在下列哪种情况下比较的次数最多。（ ）

A)  从小到大排列好的                                  B)  从大到小排列好的

C)  元素无序                                               D)  元素基本有序

(98)   对ｎ个不同的排序码进行冒泡排序，在元素无序的情况下比较的次数为（ ）。

A)  n+1             B)  n                  C)  n-1              D)  n(n-1)/2

(99)   快速排序在下列哪种情况下最易发挥其长处。（ ）

A)  被排序的数据中含有多个相同排序码

B)  被排序的数据已基本有序

C)  被排序的数据完全无序

D)  被排序的数据中的最大值和最小值相差悬殊

(100)   若一组记录的排序码为（46, 79, 56, 38, 40, 84），则利用快速排序的方法，以第一个记录为基准得到的一次划分结果为（ ）。

A)    38,  40,  46,  56,  79,  84         B)   40, 38,  46 ,  79,  56,  84

C)    40,  38，46,  56,  79,  84           D)   40,  38, 46,  84,  56,  79

(101)   下列关键字序列中，（  ）是堆。

A)   16, 72, 31, 23, 94, 53                         B)   94, 23, 31, 72, 16, 53

C)   16, 53, 23, 94，31, 72                        D)   16, 23, 53, 31, 94, 72

(101)   堆是一种（    ）排序。

A)  插入             B) 选择                 C)  交换                D)  归并

(102)   堆的形状是一棵（   ）。

A)  二叉排序树      B) 满二叉树

C)  完全二叉树      D)  平衡二叉树

(103)   若一组记录的排序码为（46, 79, 56, 38, 40, 84），则利用堆排序的方法建立的初始堆为（  ）。

A)  79, 46, 56, 38, 40, 84                            B)  84, 79, 56, 38, 40, 46

C)  84, 79, 56, 46, 40, 38                            D)  84, 56, 79, 40, 46, 38

(104)   下述几种排序方法中，要求内存最大的是（  ）。

A)  插入排序         B) 快速排序          C)  归并排序       D)  选择排序

(105)   有一组数据（15，9，7，8，20，-1，7，4），用堆排序的筛选方法建立的初始堆为（    ）。

A) -1，4，8，9，20，7，15，7

B) -1，7，15，7，4，8，20，9

C) -1，4，7，8，20，15，7，9

D) A，B，C 均不对。

(106)  下列四个序列中，哪一个是堆（ ）。

A) 75,65,30,15,25,45,20,10 B) 75,65,45,10,30,25,20,15

C) 75,45,65,30,15,25,20,10 D) 75,45,65,10,25,30,20,15

(107)   以下序列不是堆的是(    )。

A) (100,85,98,77,80,60,82,40,20,10,66)

B) (100,98,85,82,80,77,66,60,40,20,10)

C)(10,20,40,60,66,77,80,82,85,98,100)

D) (100,85,40,77,80,60,66,98,82,10,20)

(108)   快速排序方法在（   ）情况下最不利于发挥其长处。

A) 要排序的数据量太大

B) 要排序的数据中含有多个相同值

C) 要排序的数据个数为奇数

D) 要排序的数据已基本有序

(109)   对关键码序列28，16，32，12，60，2，5，72 快速排序，从小到大一次划分结果为（    ）。

A) (2,5,12,16)26(60,32,72)    B) (5,16,2,12)28(60,32,72)

C) (2,16,12,5)28(60,32,72)       D) (5,16,2,12)28(32,60,72)

(110)   对下列关键字序列用快速排序法进行排序时，速度最快的情形是（     ）。

A) {21,25,5,17,9,23,30}        B){25,23,30,17,21,5,9}

C) {21,9,17,30,25,23,5}       D) {5,9,17,21,23,25,30}

二、程序填空题

(1) 以下程序的功能是实现带附加头结点的单链表数据结点逆序连接，请填空完善之。

void reverse(pointer h)

 /\* h为附加头结点指针；\*/

{

pointer p,q;

   p=h->next;

h->next=NULL;

while((1)\_\_\_\_\_\_\_\_) {

q=p; p=p->next;

q->next=h->next;

h->next=(2)\_\_\_\_\_\_\_\_;

}

}

(2) 下面是用c语言编写的对不带头结点的单链表进行就地逆置的算法，该算法用L返回逆置后的链表的头指针，试在空缺处填入适当的语句。

void  reverse（linklist &L）

{

p=null；q=L；

while（q！=null）

{

(1)\_\_\_\_\_\_\_;

q->next=p；p=q；

(2)\_\_\_  ;

}

(3)\_\_\_\_\_;

}

(3) 以下算法的功能是用头插法建立单链表的算法，请填空完善之。

   Linklist  CreateFromHead(  )

    {      LinkList   L;

               Node   \*s;

              char     c;

              L=(Linklist)malloc(sizeof(Node));           /\*为头结点分配存储空间\*/

               \_\_\_\_\_\_\_\_\_\_\_\_\_\_;

                While( (  c=getchar())  !=’\*’  )

                {

s=(Node\*)malloc(sizeof(Node));

s->data=c;

                      \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ;

                     \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

}

        return L;

}

(4) 以下算法的功能是尾插法创建链表，请填空完善之。

typedef struct Node  /\*结点类型定义\*/

{   char data;

    struct Node  \* next;

}  Node;

Ttypedef Node\* LinkList;

Linklist  CreateFromTail(  )  /\*将新增的字符追加到链表的末尾\*/

{

LinkList L;

        Node \*r, \*s;

        char  c;

         L=(Node \* )malloc(sizeof(Node));  /\*为头结点分配存储空间\*/

        L->next=NULL;

        r=L;

       while(  (\_\_\_\_\_\_\_\_\_\_\_\_) !=’$’  )

{

s=(Node\*)malloc(sizeof(Node));

s->data=c;

               \_\_\_\_\_\_\_\_;

r = s;

}

         \_\_\_\_\_\_\_\_\_ ;

return L;

}     /\*CreateFromTail\*/

(5) 下列算法在顺序表L中依次存放着线性表中的元素，在表中查找与e相等的元素，若 L.elem[i]=e,则找到该元素，并返回i+1，若找不到，则返回“-1”  ，请填空完善之。

 int  Locate(SeqList L，int e)

{

i=0 ;

 while ((i<=L.last)&&(L.elem[i]!=e) )

\_\_\_\_\_\_\_\_\_;

      if  (\_\_\_\_\_\_\_\_\_  )

return(i+1);

      else

return(-1);

  }

(6) 下列算法在顺序表L中第i个数据元素之前插入一个元素e。 插入前表长n=L->last+1，i的合法取值范围是 1≤i≤L->last+2，请填空完善之。

void   InsList(SeqList \*L, int i,  int e)

{

 int k;

if((i<1) || (i>L->last+2))

printf(“插入位置i值不合法”)；

     if(L->last>=maxsize-1)

printf(“表已满无法插入”)；

for(k=L->last;k>=i-1;k--)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ;

   }

(7) 下列算法是在顺序表L中删除第i个数据元素，并用指针参数e返回其值。i的合法取值为1≤i≤L.last+1，请填空完善之。

 int  DelList(SeqList \*L, int i, int \*e)

{

int k;

if((i<1)||(\_\_\_\_\_\_\_\_\_\_\_\_\_\_))

printf(“删除位置不合法！”)；

         \_\_\_\_\_\_\_\_\_\_\_\_\_ ;

for(k=i;i<=L->last;k++)

       \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

       \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

}

(8) 假设以数组seqn［m］存放循环队列的元素，设变量rear和quelen分别指示循环队列中队尾元素的位置和元素的个数。

　　(1) 写出队满的条件表达式；

　　(2) 写出队空的条件表达式；

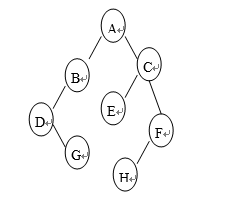
　　(3) 设m=40,rear=13,quelen=19,求队头元素的位置；

　　(4) 写出一般情况下队头元素位置的表达式。

(9) 已知一棵二叉树的中序序列为ABCDEFG，层序序列为BAFEGCD，请画出该二叉树。

(10) 已知一棵二叉树的前序序列为ABCDEFGH，中序序列为CBEDFAGH，请画出该二叉树。

(11) 已知一棵二叉树如图所示。请分别写出按前序、中序、后序和层次遍历是得到的顶点序列。



(12) 已知一棵二叉树的前序序列为：A,B,D,G,J,E,H,C,F,I,K,L中序序列：D,J,G,B,E,H, A,C,K,I,L,F。

(1)   写出该二叉树的后序序列；

(2)   画出该二叉树；

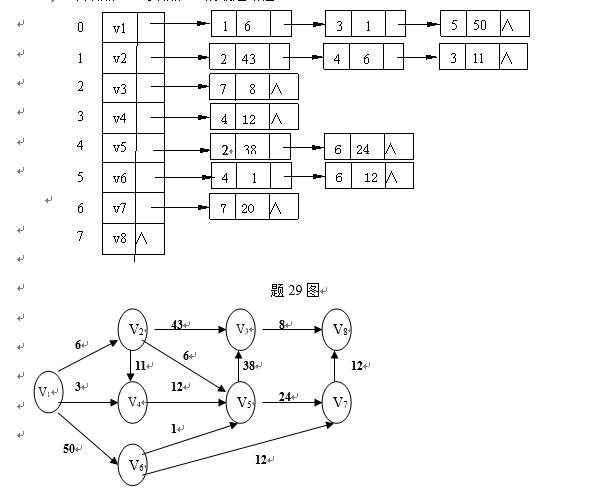
(3)   求该二叉树的高度(假定空树的高度为－1)和度为2、度为1、及度为0的结点个数。

(13)      下图是带权的有向图G的邻接表表示法。从结点V1出发，求出：

a)         深度遍历图G；

b)        G的一个拓扑序列；

c)         从结点V1到结点V8的最短路径。

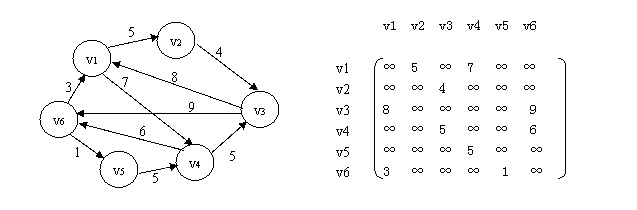


(14) 已知如图所示的有向图，请给出该图的：

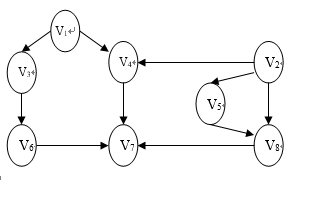
（1）       每个顶点的入度、出度；

（2）       邻接矩阵；

（3）       邻接表；



(15) 对下面的有向图，从顶点V1开始进行遍历，试画出遍历得到的DFS生成森林和BFS生成森林。



三、算法设计题(10分)

(1) 阅读下列递归算法，写出非递归方法实现相同功能的C程序。

void  test(int  &sum)

 { int  x；

scanf(x)；

if(x=0) sum=0

else {test(sum); sum+=x;}

printf(sum)；

}

(2) 试写出把图的邻接矩阵表示转换为邻接表表示的算法。

设图的邻接矩阵为g[n][n]（针对无向图）

(3) 阅读算法并根据输入数据画出链表。

linklist createlistr1(  )

{    char  ch;

     linklist      head=(listnode\*)malloc(sizeof(listnode));

     listnode  \*p，\*r；

     r=head;

     while((ch=getchar( ))!=‵\n′)

     {     p=(listnode\*)malloc(sizeof(listnode));

           while  (p)  p=(listnode\*)malloc(sizeof(listnode));

           p–>data=ch;       r–>next=p;         r=p;

     }

     r–>next=NULL;

     return(head);

}

输入数据为：text8

(4) 阅读算法并指出下列各段程序完成的功能。

void  add\_poly(Lnode \*pa,Lnode \*pb)

{

Lnode \*p,\*q,\*u,\*pre;   int x;

   p=pa->next;   q=pb->next;   pre=pa;

   while((p!=NULL) && ((q!=NULL)) {

if (p->exp < q->exp)

           { pre=p;  p=p->next;       }

       else  if (p->exp= =q->exp)

               {

x=p->coef+q->coef;

if  (x!=0)

{ p->coef=x;  pre=p;}

                     else

{ pre->next=p->next;  free(p); }

                     p=pre->next;    u=q;

q=q->next; free(u);

                }

               else

               {

u=q->next;q->next=p;pre->next=q;

pre=q;  q=u;

}

}

if (q!=NULL)   pre->next=q;

     free(pb);

}

(5) 阅读下面的程序，说明程序的具体功能。

typedef int elementype

typedef struct node

{

elemtype  data;

strunct node  \*next;

}linklist;

void function( linklist \*head, elemtype x )

{

linklist \*q, \*p;

q=head;

p=q-next;

while (p !=NULL) && (p->data != x )

{  q=p;  p=p->next;  }

if  (q==NULL)

printf (“there is no this node ::\n”);

else

{  q ->next =p ->next ;  free (p);    }

}

(6) 阅读下面的程序，说明程序的具体功能。

void function(  ) {

initstack(s);

              scanf (“%”,n);

              while(n) {

push(s,n%8);         n=n/8;

}

              while(! Stackempty(s)) {

pop(s,e);

printf(“%d”,e);

}

        }

(7) 阅读下面的程序，说明程序的具体功能。

void    print(int  w)

{

int  i;

       if ( w!=0)

       {

print(w-1);

             for(i=1;i<=w;++i)

                  printf(“%3d,”,w);

             printf(“/n”);

       }

}

运行结果：

1，

2，2，

3，3，3，

(8) 阅读下面的程序，分别说明程序中四个for循环和语句 ++cpot[col];的具体功能。

void FastTransposeSMatrix(Matrix M, Matrix &T)

{

T.mu=M.nu;  T.nu=M.mu;  T.tu=M.tu;

        if (T.tu)

         {

for (col=1;col<=M.nu;++col)

num[col]=0;

for (t=1;t<=M.tu;++t)

++num[M. item[t].j];

              cpot[1]=1;

for (col=2;col<=M.nu;++col)

                              cpot[col]=cpot[col-1]+num[col-1];

               for (p=1;p<=M.tu;++p)

                {

col=M. item[p].j;

                     q=cpot[col];

T.item[q].i=M.data[p].j;

                     T.item [q].j=M. item[p].i;

T.item[q].e=M.data[p].e;

   ++cpot[col];

                 }

}

return OK;

}

(9) 已知二叉树的二叉链表存储表示，指出建立如图所示树的结构时输入的字符序列。

typedef struct BiTNode

{    char data;

     struct BiTNode \*lchild,\*rchild;

} BiTNode;

void CreateBiTree( BiTNode \*T)

{

char ch;

    scanf(&ch);

    if(ch= =‘ ‘) T=NULL; //‘’中间是空格

    else{

T=(BiTNode \*)malloc(sizeof(BiTNode));

while (T==NULL)

T=(BiTNode \*)malloc(sizeof(BiTNode));

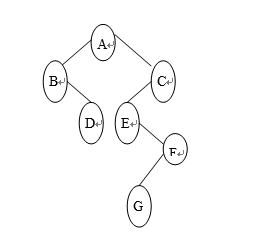
T–>data=ch;

                 CreateBiTree(T–>lchild);

                 CreateBiTree(T–>rchildd);

           }

}



(10) 已知二叉树的二叉链表存储表示，写出中序遍历的递归算法。

typedef struct BiTNode

{    char data;

     struct BiTNode \*lchild,\*rchild;

} BiTNode,\*BiTree;

(11) 阅读下面的程序，分别指出程序中三个for循环、整个程序以及语句scanf("%d,%d",&G.vexnum,&G.arcnum);  的功能。

void funcgraph(MGraph &G)

{     int i,j,k,w;      char v1,v2;

       printf("Input vexnum & arcnum:");

       scanf("%d,%d",&G.vexnum,&G.arcnum);

       printf("Input Vertices:");

       for (i=0;i<G.vexnum;i++)

         scanf("%c",&G.vexs[i]);

       for (i=0;i<G.vexnum;i++)

              for (j=0;j<G.vexnum;j++)

                     G.arcs[i][j]=0;

for (k=0;k<G.arcnum;k++)

    {     printf("Input Arcs(v1,v2 & w):\n");

              scanf("%c%c,%d",&v1,&v2,&w);

              i=LocateVex(G,v1);

              j=LocateVex(G,v2);

              G.arcs[i][j]=w;   G.arcs[j][i]=w;

    }

}