一、选择题

1、从逻辑结构上可以把数据结构分为【 C 】。

A、动态结构和静态结构　　B、紧凑结构和非紧凑结构

C、线性结构和非线性结构　　D、内部结构和外部结构

2、在一个单链表中，若p所指结点不是表尾结点，在p之后插入s所指结点，则执行【 B 】。

A、s->next = p; p->next = s;　B、s->next = p->next; p->next = s;

C、s->next = p->next; p = s;　C、p->next = s; s->next = p;

3、链表结构不具有下列【 B 】特点。

A、插入和删除无需移动元素　　B、可随机访问链表中的任意元素

C、无需实现分配存储空间　　D、所需空间与结点个数成正比。

4、在一个单链表中，已知q所指结点是p所指结点的前驱结点，若在q和p之间插入s结点，则执行【 C 】。

A、s->next = p->next; p->next = s;

B、p->next = s->next; s->next = p;

C、q->next = s; s->next = p;

D、p->next = s; s->next = q;

5、一个栈的入栈序列是1，2，3，4，5，则栈不可能输出的序列是【 C 】。

A、54321　　B、45321　　C、43512　　D、12345

6、判断一个队列Q（元素最多为M个）为空的条件是【 C 】。

A、Q->rear - Q->front = M　　B、Q->rear - Q->front -1 ==M

C、Q->rear == Q->front　　D、Q->rear + 1 == Q->front

7、在一个链队列中，假设f和r分别指向队首和队尾，则插入s所指结点的运算是【 A 】。

A、r->next = s; r=s;　　B、r->next = s; f=s;

C、s->next = r; r=s;　　D、s->next = f; f=s;

8、深度为5的二叉树至多有【 A 】个结点。

A、31　　B、32　　C、16　　D、10

9、在一非空二叉树的中序遍历序列中，根结点的右边【 A 】。

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、只有左子树上的所有结点　　D、只有左子树上的部分结点

10、如果一棵完全二叉树有1001个结点，则其叶子结点个数为【 D 】。

A、250　　B、500　　C、502　　D、490

11、在一个图中，所有顶点的度数之和是所有边数的【 C 】倍。

A、1/2　　B、1　　C、2　　D、4

12、采用邻接表存储的图的深度优先遍历算法类似于二叉树的【 A 】。

A、先序遍历　B、中序遍历　C、后序遍历　D、按层遍历

13、一个有n个顶点的无向图最多有【　　】条边。

A、n　　B、n(n-1)　　C、2n　　D、n(n-1)/2

14、静态查找表与动态查找表的根本区别在于【　　】。

A、它们的逻辑结构不同　　B、施加在其上的操作不同

C、所包含的数据元素类型不同　D、存储实现不一样

15、顺序查找适用于存储结构为【　　】的线性表。

A、哈希存储　　B、压缩存储

C、顺序存储或链式存储　　D、索引存储

16、若一颗二叉树的先序遍历序列与后序遍历序列正好相反，则该二叉树一定满足【　　】。

A、所有结点均无孩子　　B、所有结点均无右孩子

C、只有一个叶子结点　　D、是一颗满二叉树

17、二叉排序树是【　　】。

A、每一分支结点的度均为2的二叉树

B、中序遍历得到一升序序列的二叉树

C、按从左到右顺序编号的二叉树

D、每一分支结点的值均小于左子树上所有结点的值，又大于右子树上所有结点的值

18、具有12个记录的序列，采用冒泡排序最少的比较次数是【　　】。

A、1　B、144　　C、11　　D、66

19、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

20、在一个包含n个顶点e条边的无向图的邻接矩阵中，零元素的个数为【　　】。

A、e　　B、2e　　C、n^2-e　　D、n^2-2e

二、判断对错

【　　】1、具有n个顶点的连通图至少有n条边。

【　　】2、链表的单个结点内部的存储空间可以是不连续的。

【　　】3、栈和队列的共同点是只允许在端点处插入和删除元素。

【　　】4、使用循环队列可以解决队列顺序存储时的假溢出问题。

【　　】5、要想通过遍历序列还原为惟一二叉树，应当知道其先序序列和后序序列。

【　　】6、若一个结点是某二叉树子树的中序遍历序列的第一个结点，则它也必是该子树的后序遍历序列的第一个结点。

【　　】7、完全二叉树可采用顺序存储结构存储，非完全二叉树则不能。

【 √ 】8、对于一棵含有n个结点的完全二叉树，将其结点按从上到下且从左至右按1至n进行编号，则对其任意一个编号为i的结点，如果它有左孩子，则其左孩子结点的编号为2i。

【 √ 】9、哈夫曼树的所有子树也都是哈夫曼树。

【 × 】10、当图的边较少而结点较多时，求其最小生成树用Prim算法比用Kruskal算法效率更高。

三、 填空题

1、向量的第一个元素的存储地址是200，每个元素的长度是3，那么第6个元素的存储地址是 【1】215 。

2、在一个带头结点的单链表中，p 所指结点既不是首元结点，也不是尾元结点，删除 p 结点的语句序列是 【2】 、 【3】 、 【4】 。

3、设堆栈有足够的存储空间，那么向堆栈中插入一个数据元素，即入栈的操作过程是 【5】 、 【6】 。

4、一般情况下，向循环队列中插入数据元素时，需要判满队列是否已经满了，判断条件是： 【7】 。

5、已知循环队列用数组data[1...n]存储元素值，front和rear分别表示队头和队尾指针，则当前队列中元素的个数为 【8】 。

6、深度为k的二叉树最多有 【9】 个结点，深度为k的完全二叉树最少有 【10】 个结点（k≥1）。

7、如以{2, 3, 6, 7, 9}作为叶子结点的权值构造哈夫曼树，则其最短带权路径长度为 【11】 。

8、已知某二叉树的中序序列和前序序列分别为42758136、12457836，则它的后序序列为 【12】 。

9、在有n个顶点的有向图中，每个顶点的度最大可达到 【13】 。

10、在有序表A[1...18]中，采用折半查找算法查找元素值等于A[7]的元素，所比较过的元素的下标依次为 【14】 。

11、一组记录的输入顺序为(25, 38, 65, 90, 72, 14)，则利用堆排序方法建立的初始“小顶堆”为 【15】 。

四、简答题

1、设有一段正文是由字符集{a, b, c, d, e, f, g, h}组成，正文长度为 89 个字符，其中每个字符在正文中出现的次数分别为 17, 12, 14, 4, 10, 9, 20, 3。若采用哈夫曼树对这段文字进行压缩存储，请完成如下工作：

（1）构造哈夫曼树（规定权值较小的结点为左子树）；

（2）求出每个字符的哈夫曼编码；

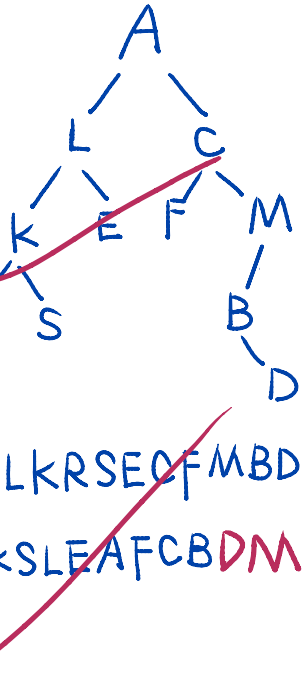
2、一棵有 11 个结点的二叉树的存储情况如下图所示（其中“∧”表示空指针），left[i]和 right[i]分别表示结点 i 的左、右孩子，根结点是序号为 3 的结点，要求：

（1）画出该二叉树；

（2）分别写出该二叉树的前序和中序遍历序列。

结点编号 i	1	2	3	4	5	6	7	8	9	10	11
LeftChild[i]	6	∧	7	∧	8	∧	5	∧	2	∧	∧
Data[i]	M	F	A	D	K	B	L	R	C	S	E
RightChild[i]	∧	∧	9	∧	10	4	11	∧	1	∧	∧

第 2 题图



前序 ALKRSECFMBD

中序 RKSLEAFCBDM

3、设数据集合 D={2, 24, 12, 15, 32, 9, 10, 35, 7, 5}，要求：

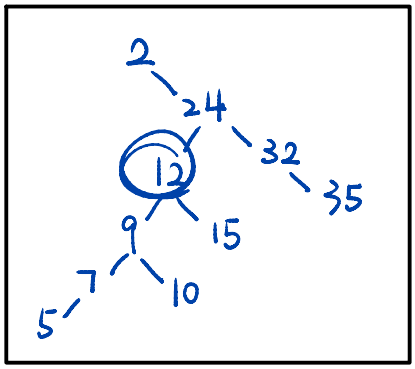
（1）依次读取 D 中的各个数据，构造一棵二叉排序树 Bt；

（2）如何根据此二叉树 Bt 求得数据集合 D 的一个有序序列？并写出该有序序列；

（3）画出在上述二叉树中删除结点“12”后得到的二叉树结构。

4、用深度优先和广度优先遍历算法对下图 G 进行遍历（要求从顶点 A 出发），请给出深度优先和广度优先遍历序列。

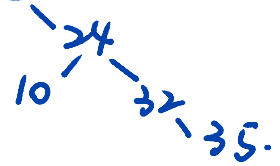
3(1)



(2) 中序遍历

2, 5, 7, 9, 10, 12, 15, 24, 32, 35

(3)

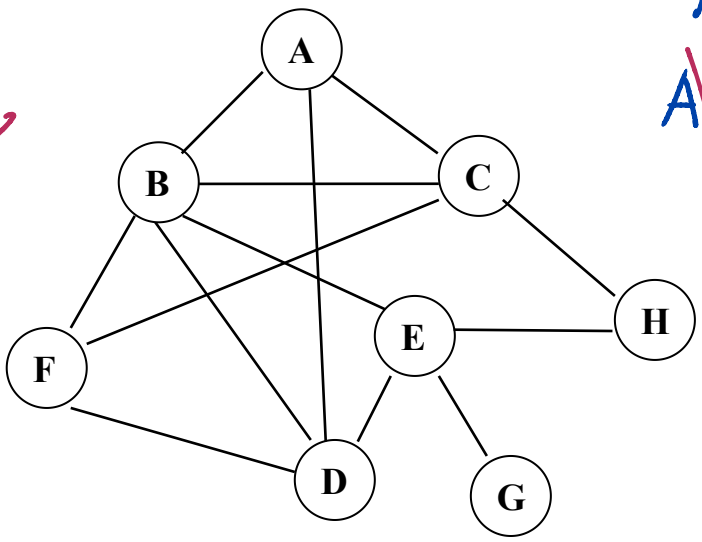


深度优先 DFS.

ABFDEGHC.

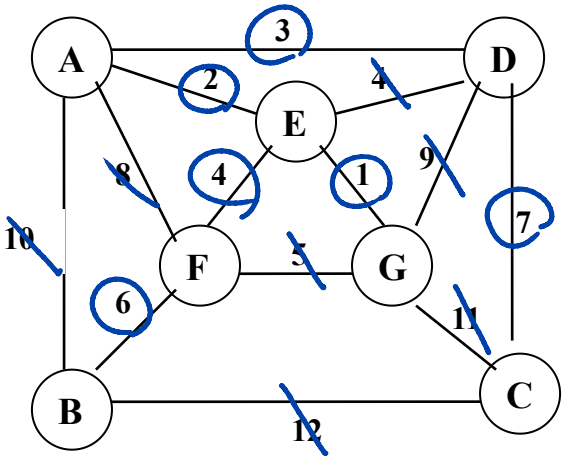
广度优先 BFS.

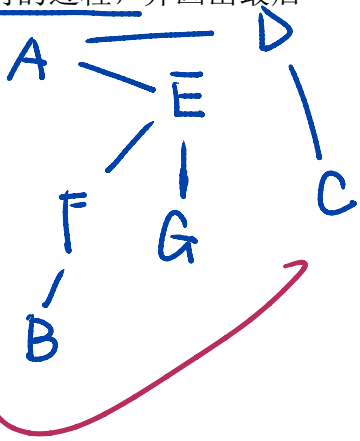
ABCFDEHG.



第 4 题图

5、对于如下所示的加权无向图，写出用 Prim 算法构造最小生成树的过程，并画出最后得到的最小生成树。



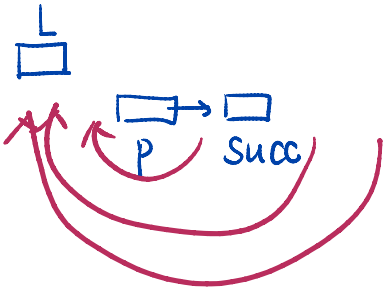


第 5 题图

五、按照指定功能，完成下列算法

1、逆置带头结点的单链表 L

```
void  inverse(LinkList &L) {
    p=L->next;  L->next=NULL;
    while ( p) {
        succ=p->next;
        【1】 P->next=L->next
        【2】 L=p;
        p = succ;
    }
}
```



2、算术表达式求值的算符优先算法。设 OPTR 和 OPND 分别为运算符栈和运算数栈，OP 为运算符、界限符集合。

```
operandType EvaluateExpression( )
{
     InitStack(OPTR);          Push (OPTR, #);
     InitStack(OPND);          c=getchar( );
     while (【3】_____)
     {
       if  (! In (c, OP))
       {
             【4】_____
             c=getchar( );
        }
        else
            switch ( 【5】_____ )
            {
             case <:
                 【6】_____   c=getchar( );      break;
             case =:
                 【7】_____      c=getchar( );      break;
             case >:
                  Pop ( OPTR,  theta);
                  Pop ( OPND, b);        Pop(OPND, a);
                  【8】_____
                  break;
            } //switch
    } //while
    【9】_____
} //EvaluateExpression
```

3、中序遍历递归算法

```
void InOrderTraverse ( BiTree T , Status ( * Visit ) ( ElemType  e ) )
{    // 采用二叉链表存贮二叉树， visit( )是访问结点的函数
      // 本算法中序遍历以 T 为根结点指针的二叉树
   if ( T )
   {
      【10】InorderTraverse ( T→Lchild, visit )
      【11】visit (T) T→data
      【12】InorderTraverse ( T-Rchild, visit )
   }
} //InOrderTraverse
```

4、在有序表 ST 中折半查找法查找其关键字等于 key 的数据元素。若找到，则返回该元素在表中的位置，否则为 0。

```
int Search_Bin ( SSTable ST, KeyType key )
{
   low = 1;  high = ST.length;
   while (【13】) {
      mid = (low + high) / 2;
      if  (EQ (key , ST.elem[mid].key) )
```

```
        return  mid;
      else  if ( LT (key , ST.elem[mid].key) )
        【14】
      else  【15】
   }
   return 0;
} // Search_Bin
```

六、给出下列算法的功能描述或程序运行结果

（一）、请描述算法的功能

1、（4 分）

```
typedef struct node{
datatype data;
struct node *link;
} *LinkList;
int Algo(LinkList list)
{
if(list==NULL)
return 0;
else
return 1+Algo(list->link);
}
```

计算线性表长度

2、（4 分）

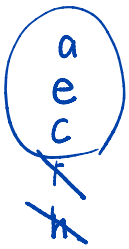
```
void Algo(adjlist g)
{
    int i，j，k;
    struct vexnode *s;
    for (k=1;k<=n;k++)
    {
        g[k].data=k;
        g[k].link=NULL;
    }
    printf("输入一个偶对(弧尾和弧头):");
    scanf("%d，%d"，&i，&j);
    while (i!=0 && j!=0)
    {
        s=(struct vexnode *)malloc(sizeof(vexnode));
        s->adjvex=j;
        s->next=g[i].link;
        g[i].link=s;
        printf("输入一个偶对(弧尾和弧头):");
        scanf("%d，%d"，&i，&j);
    }
}
```

（二）、请给出程序的运行结果

3、（4 分）

```
void main()
{
```

```
    Queue Q; InitQueue(Q);
    char x='e', y='c';
    EnQueue(Q,'h'); EnQueue(Q,'r'); EnQueue(Q,y);
    DeQueue(Q,x); EnQueue(Q,x);
    DeQueue(Q,x); EnQueue(Q,'a');
    while(!QueueEmpty(Q))
    {
        DeQueue(Q,y);
        printf(y);
    }
    printf(x);
}
```



4.（3 分）

```
#define N 4
void main( )
{
    SqQueue q;              //定义一个顺序队列 q
    int i,j,e,pre=N,curgroup=0,num=0;
    int allclash[N][N]={{0,1,1,0},{1,0,1,0},{0,0,0,0},{1,1,0,1}};
    int clash[N], group[N];

     InitQueue (&q);       //初始化队列
    for(i=0;i<N;i++)
        EnQueue ( &q, i );//将 i 入队

    while(!QueueEmpty(q)&&num<N)
    {   DeQueue ( &q, &e );// 删除队头元素，用 e 返回队头元素值
        if ( e <=pre )        // 开辟新的组
        {
            curgroup++;
            for(i=0;i<N;i++)       clash[i]=0;
        }
        if ( clash[e]==0 )            //e 能入组
        {
            group[e]=curgroup;         //e 入组，记下序号为 i 的元素所属组号;
            for(i=0;i<N;i++)       //修改 clash 数组;
                clash[i]=clash[i]+allclash[e][i];
            num++;
        }
        else            EnQueue ( &q, e );          //e 重新入队列;
        pre=e;
    }

for(i=1;i<=curgroup;i++)
    {   printf("group %d :",i);
        for(j=0;j<N;j++)
            if(group[j]==i)          printf("%d ",j);
        printf("\n");
    }
}
```