请将答案撰写在答题纸上！

校内考试结束后，请将试卷和答题纸一并上交！

一. 选择题（30 分，每小题 2 分）

1. 下列叙述中正确的是（　　）。

A: 散列是一种基于索引的逻辑结构

B: 基于顺序表实现的逻辑结构属于线性结构

C: 数据结构设计影响算法效率，逻辑结构起到了决定作用

D: 一个逻辑结构可以有多种类型的存储结构，且不同类型的存储结构会直接影响到数据处理的效率

2. 在广度优先搜索算法中，一般使用什么辅助数据结构？（　　）

A: 队列　　B: 栈

C: 树　　D. 散列

3. 若某线性表实现采取链式存储，那么该线性表中结点的存储地址（　　）。

A: 一定不连续　　B: 既可连续亦可不连续

C: 一定连续　　D: 与头结点存储地址保持连续

4. 若某线性表的常用操作是在表尾插入或删除元素，则时间开销最小的存储方式是（　　）。

A: 单链表　　B: 仅有头指针的单循环链表

C: 顺序表　　D. 仅有尾指针的单循环链表

5. 给定后缀表达式（逆波兰式）ab+-c*d-，其对应的中缀表达式是（　　）。

A: a-b-c*d　　B: -(a+b)*c-d

C: a+b*c-d　　D: (a+b)*(-c-d)

6. 今有一空栈 S，基于待进栈的数据元素序列 a, b, c, d, e, f，依次进行进栈、进栈、出栈、进栈、进栈、出栈的一系列操作，上述操作完成以后，栈 S 的栈顶元素为（　　）。

A: f　　B: c　　C: a　　D: b

7. 对于带头结点的单链表，其表头为 head，则判定空表的条件是(　　)。

A: head==NULL　　B: head!=NULL

C: head→next==head　　D: head→next==NULL

8. 以下典型排序算法中，具有稳定排序特性的是（　　）

A: 冒泡排序（Bubble Sort）　　B: 直接选择排序（Selection Sort）

C: 快速排序（Quick Sort）　　D: 希尔排序（Shell Sort）

9. 以下关键字序列中，可以有效构成一个大根堆（即最大值堆，最大值在堆顶）的序列是（　　）。

A: 5 8 1 3 9 6 2 7　　B: 9 8 1 7 5 6 2 33

C: 9 8 6 3 5 1 2 7　　D: 9 8 6 7 5 1 2 3

10. 以下典型排序算法中，内存开销最大的是（　　）。

A: 冒泡排序　　B: 快速排序

C: 归并排序　　D: 堆排序

11. 排序算法往往依赖于对待排序元素序列的多趟比较/移动操作（即执行多轮循环），第一趟结束后，任一元素都无法确定其最终排序位置的算法是 （　　）。

A: 选择排序　　B: 快速排序

C: 冒泡排序　　D: 插入排序

12. 考察以下基于单链表的操作，相较于顺序表实现，带来更高时间复杂度的操作是（　　）。

A. 合并两个有序线性表，并保持合成后的线性表依然有序

B. 交换第一个元素与第二个元素的值

C. 查找某一元素值是否在线性表中出现

D. 输出第 i 个（0<=i<n, n 为元素个数）元素

13. 已知一个整型数组中的元素值依次为{19，20，50，61，73，85，11，39}，采用某种排序算法，在多趟比较/移动操作（即执行多轮循环）后，依次得到如下中间结果（每一行对应一趟）：

（1）19 20 11 39 73 85 50 61

（2）11 20 19 39 50 61 73 85

（3）11 19 20 39 50 61 73 85

请问，上述过程使用的排序算法是（　　）算法。

A. 冒泡排序　　B. 插入排序　　C. 希尔排序　　D. 归并排序

14. 今有一非连通无向图，共有 36 条边，该图至少有（　　）个顶点。

A: 8　　B: 9　　C: 10　　D: 11

15. 令 G=(V, E) 是一个无向图，若 G 中任何两个顶点之间均存在唯一的简单路径相连，则下面说法中错误的是（　　）。

A: 图 G 中添加任何一条边，不一定造成图包含一个环

B: 图 G 中移除任意一条边得到的图均不连通

C: 图 G 的逻辑结构实际上退化为树结构

D: 图 G 中边的数目一定等于顶点数目减 1

二. 判断（10 分，每小题 1 分；对填写“Y”, 错填写“N”）

1. （　　）按照前序、中序、后序方式周游一棵二叉树，分别得到不同的结点周游序列，然而三种不同的周游序列中，叶子结点都将以相同的顺序出现。

2. （　　）构建一个含 N 个结点的（二叉）最小值堆，时间效率最优情况下的时间复杂度大 O 表示为 O (N Log N)。

3. （　　）对任意一个连通的无向图，如果存在一个环，且这个环中的某一条边的权值不小于该环中任意一个其它的边的权值，那么这条边一定不会是该无向图的最小生成树中的边。

4. （　　）通过树的周游可以求得树的高度，若采取深度优先遍历方式设计求解树高度问题的算法，算法空间复杂度大 O 表示为 O（树的高度）。

5. （　　） 树可以等价转化二叉树，树的先序遍历序列与其相应的二叉树的前序遍历序列相同。

6. （　　） 如果一个连通无向图 G 中所有边的权值均不同，则 G 具有唯一的最小生成树。

7. （　　） 求解最小生成树问题时，Kruskal 算法更适用于稀疏图，Prim 算法更适用于稠密图。

8. （　　） 使用线性探测法处理散列表碰撞问题，若表中仍有空槽（空单元），插入操作一定成功。

9. （　　）从链表中删除某个指定值的结点，其时间复杂度是 $O(1)$。

10. （　　）弗洛伊德（Floyd）算法是一种求解有向和无向图最短路径的动态规划算法，但该算法的局限性是无法正确求解带有负权值边的图的最短路径。

三． 填空（20 分，每题 2 分）

1. 定义二叉树中一个结点的度数为其子结点的个数。现有一棵结点总数为 101 的二叉树，其中度数为 1 的结点有 30 个，则度数为 0 结点有_____个。

2. 定义完全二叉树的根结点所在层为第一层，如果一个二叉树的第六层有 23 个叶结点，则它的总结点数量可能有_____个。（请填写所有 3 个可能的结点数，写对 1 个得 1 分，2 个得 1.5 分，写错 1 个不得分）。

3. 对于初始排序码序列（51，41，31，21，61，71，81，11，91），第 1 趟快速排序（以第一个数字为 pivot 轴值）的结果是：_____。

4. 如果输入序列是已经正序，在（改进）冒泡排序、直接插入排序和直接选择排序算算法中，_____算法最慢结束。

5. 已知某二叉树的先根周游序列为{ A，B，D，E，C，F，G }，中根周游序列为{ D，B，E，A，C，G，F }，则该二叉树的后根次序周游序列 {_____ }。

6. 使用栈计算后缀表达式 （操作数均为一位数）“1 2 3 + 4 * 5 + 3 + –”，当扫描到第二个+号但还未对该+号进行运算时，栈的内容 （以栈底到栈顶从左往右的顺序书写） 为_____。

7. 51 个顶点的连通图 G 有 50 条边，其中权值为 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 的边各 5 条，则连通图 G 的最小生成树各边的权值之和为_____ 。

8. 包含 n 个顶点无向图的邻接表存储结构中，所有顶点的边表中最多有_____个结点。具有 n 个顶点的有向图，一个顶点入度和出度之和最大值不超过_____。

9. 给定一个长度为 7 的空散列表，采用双散列法解决冲突，两个散列函数分别为：h1(key)=key%7，h2(key)=key%5+1 请向散列表依次插入关键字为 30,58,65 的集合元素，插入完成后 65 在散列表中存储地址为_____。

10. 阅读算法 ABC，回答问题。

```
int ABC (int n )
{    int k = 1, m = (int) sqrt (n) ;   // sqrt(n) 用来求给 n 的平方根
     While ( (++k <= m) && (n%k != 0) );
     return (k > m) ? 1 : 0;
}
```

1) 算法的功能是：_____ 。

2) 算法的时间复杂度是 O(_____)。

四. 简答（3 题，共 14 分）

1. （4 分）字符串匹配算法从长度为 n 的文本串 S 中查找长度为 m 的模式串 P 的首次出现位置。
   a) 字符串匹配的朴素算法使用暴力搜索，大致过程如下：对于 P 在 S 中可能出现的 n-m+1 个位置，比对此位置时 P 和 S 中对应子串是否相等。其时间复杂度 O((n-m+1)*m)。请举例说明算法时间复杂度一种最坏情况（注：例子中请只出现 0 和 1 两种字符）。（1 分）
   b) 已知字符串 S 为“abaabaabacacaabaabcc”，模式串 t 为“abaabc”。采用 KMP 算法进行查找，请写出 next 数组。（2 分）
   c) 说明在上述(b)步骤中，第一次出现不匹配（s[i] != t[j]）时，i=j=5；则下次比对时，i 和 j 的值分别是？（1 分）

2. （5 分）有八项活动，每项活动标记为 V+编号 n(0<=n<=7)，每项活动要求的前驱如下:

活动	V0	V1	V2	V3	V4	V5	V6	V7
前驱	无前驱	V0	V0	V0, V2	V1	V2, V4	V3	V5, V6

试画出相应的 AOV 网络，并给出一个拓扑排序序列，如存在多种，则按照编号从小到大排序，输出最小的一种。

3. （5 分）简要回答下列 Huffman 树以及 Huffman 编码的相关问题。
   a) 假定需要编码的字符个数为 m，哈夫曼树构造后结点总数多少？简要阐述理由（1 分）
   b) 假定报文中各字符及其出现次数为：A（50），B（30），C（10），D（25），E（11），F（99），G（8），H（51），I（22）。试画出对应的 Huffman 树（严格按照左小右大构造），并给出各个字符的 Huffman 编码（左分支编码 0，右分支编码 1）。（4 分）

五. 算法填空（4 题，共 26 分）

1. （6 分）请填空完成下列程序：读入一个从小到大排好序的整数序列到链表，然后在链表中删除重复的元素，使得重复的元素只保留 1 个，然后将整个链表内容输出。（为简单起见程序不释放动态分配的空间）

输入：第一行是正整数 n (n < 80)，第二行是 n 个整数
输入样例：
9
1 2 2 2 3 3 4 4 6
输出样例:
1 2 3 4 6

```
#include <stdio.h>
#include <malloc.h>
struct Node {
   int data;
   struct Node * next;
};
int main() {
   int n,i, a[100];
   scanf("%d",&n);
   for(i = 0;i < n; ++i) scanf("%d",a+i);
   struct Node * head = (struct Node *)malloc(sizeof(struct Node));
   head->data = a[0];  head->next = NULL;
   struct Node * p = head;
   for( i = 1;i < n; ++i) { //建链表的过程
      p->next = (struct Node *)malloc(sizeof(struct Node));
      ____________ = a[i] ;          ①      //1 分
      ____________________;          ②      //1 分
      p = p->next;
   }
   p = head;
   while (p) {   //删除过程
      while( (_____③_____) && (p->data == p->next->data))     //2 分
      ____________________________; ④          //1 分
         p = p->next;
   }
   p = head;
   while (p) {
      printf("%d   ",p->data);
      ____________________;    ⑤    //1 分
   }
   return 0;
}
```

2. （7 分）为实现对一组数据根据排序码进行自小到大排序，请填空完成下列堆排序程序（仅含关键部分代码）。程序中建立的堆是大根堆（即最大值堆，最大元素在堆顶）。

```
typedef struct   {
     int key;                /*  排序码字段  */
     float info;             /*  其它字段  */
}RecordNode;
typedef struct   {
     RecordNode   record [100];
     int n;                          // n 记录个数，0≤n≤100
}SortObject;

void sift(SortObject *p, int i, int n)          //  调整过程
{
     int child;
     RecordNode temp =p->record[i];
     child =      ①          ;       //1 分       /*child 是 record[i]左子女*/
     while( child <n )
     {
          if( (          ②     ) && (p->record[child].key<p->record[child+1].key) )    //1 分
               child++;
          if( temp.key < p->record[child].key)   {        /*进入下一层继续调整*/
                              ③     ;                    //2 分
               i=child;
               child=                ④ ;                  //1 分
          }
          else break;                /*调整结束*/
     }
     p->record[i]=temp ;   /*将记录 Ri 放入正确位置*/
}

void heapSort( SortObject *pvector)   {     //堆排序
     int i,n;     RecordNode temp;
     n=pvector->n;
     for( i=n/2-1; i>0; i-- )        /*  建立初始堆*/
          ⑤      ;                  //1 分
     for (i=n-1;i>0;i--)            /*进行 n-1 趟堆排序*/
     {
          temp=pvector->record[0];                /*  当前堆顶和最后记录互换*/
          pvector->record[0]=pvector->record[i];
          pvector->record[i]=temp;
          ⑥       ;                  //1 分
     }
}
//  以下略
```

3. （7 分）以下代码实现了对 AOV 网的拓扑排序，请阅读代码并补全空缺。

```
#include <stdio.h>
#include <stdlib.h>
struct EdgeNode;
typedef struct EdgeNode * PEdgeNode;
typedef struct EdgeNode * EdgeList;
struct EdgeNode {
   int endvex;          /* 相邻顶点在顶点表中的下标 */
   AdjType weight;      /* 边的权重 */
   PEdgeNode nextedge;
};
typedef struct{
   int vertex;          /* 存储顶点信息 */
   EdgeList edgelist;   /* 边表头指针 */
} VexNode;
typedef struct{
   int n;               /* 图的顶点个数 */
   VexNode *vexs;       /*顶点表 */
}GraphList;

void findInDegree(GraphList* g, int *indegree)      /*求所有顶点的入度 */
{
   int i;
   PEdgeNode p;
   for(i=0; i<g->n; i++)  indegree[i] = 0;
   for(i=0; i<g->n; i++) {
        p = g->vexs[i].edgelist;
        while(p) {
            ①       ;                    //1 分
           p = p->nextedge;
       }
   }
}

int topoSort(GraphList * paov, int * ptopo)              /*拓扑排序 */
{
   EdgeList p; int i, j, k, nodeno=0, top=-1;
   int *indegree=(int*)malloc(sizeof(int)*paov->n);
   findInDegree(paov, indegree);
   for(i=0; i<paov->n; i++)
        if(indegree[i]==0) {                    /*将入度为零的顶点入栈（栈直接用 indegree 表示）*/
                ;      ②       //1 分
                top=i;
        }
```

```
    while( ③ )        /*栈不为空 */          //1 分
    {
        j=top;
        top=indegree[top];
        ptopo [nodeno++ ] = j;
        p= ____________;  ④          //1 分
        while(p)
        {
            k=____________;  ⑤          //1 分
            indegree[k] = ____________;  ⑥          //1 分
            if(indegree[k]==0) {   indegree[k]=top;   top=k;    }
            p=p->nextedge;
        }
    }
    free(indegree);
    if(____________⑦ )          //1 分
    {
        printf("The aov network has a cycle\n");
        return(0);
    }
    return(1);
}
////////////////////////////////////////
int main() {
     GraphList * paov;
     /* 以下代码省略，功能为读入 int 型数据构建图，并用邻接表表示法存储 */
       …
     /* 建图完毕后，用指针 paov 指向构建的图*/
    If (paov->n==0)     return (0)；
    int *ptopo = (int*)malloc(sizeof(int)* paov->n);     /*排序结果存放在 ptopo 中*/
    if (toptSort(paov, ptopo))
    {
       for(int i=0;i<paov->n;i++)     printf("%d", ptopo[i]);
       free(ptopo);              /* 释放数组 ptopo 占用的内存 */
       destorygraph(paov);    /* 释放图 paov 占用的内存 */
       return(1);
    }
   else
    {
       free(ptopo);         /* 释放数组 ptopo 占用的内存 */
       destorygraph(paov);    /* 释放图 paov 占用的内存 */
       return (0);
     }
}
```

4. （6 分）请填空完成下列程序：给定一个无向图，判断是否连通，是否有回路。

输入: 第一行两个整数 n,m，分别表示顶点数和边数。顶点编号从 0 到 n-1 (1<=n<=110, 1<=m <= 10000)
接下来 m 行，每行两个整数 u 和 v，表示顶点 u 和 v 之间有边。
输出:
如果图是连通的，则在第一行输出“connected:yes",否则第一行输出“connected:no"。
如果图中有回路，则在第二行输出“loop:yes ",否则第二行输出“loop:no"。

样例输入
3 2
0 1
0 2
样例输出
connected:yes
loop:no

```
#include <stdio.h>
#include <memory.h>
#define maxN 120
int G[maxN][maxN], n,m, visited[maxN];   //G 是邻接矩阵
int total = 0;
void dfsConnection(int v) {//从 v 出发深度优先遍历图
   visited[v] = 1;
   total += 1;
   int u;
   for(u = 0;u < n; ++u) {
      if (G[v][u] && !visited[u])      dfsConnection(u);
   }
}

int dfsLoop(int v,int x)   //深度优先遍历图寻找环 ， x 是深度优先搜索树上 v 的父结点
{
   visited[v] = 1;     int u;
   for(u = 0;u < n; ++u)
   {
      if ( G[v][u] )
      {
         if (!visited[u])
            if (_____①_____)     return 1;        //1 分
         else
            if (_____②_____)          return 1;        //2 分
      }
   }
   return 0;
}
```

```
int main()
{
   scanf("%d%d",&n,&m);
   int i;
   memset(G,0,sizeof(G));
   memset(visited,0,sizeof(visited));
   for(i = 0; i< m; ++i) {
      int a,b;
      scanf("%d%d",&a,&b);
      G[a][b] = G[b][a] = 1;
   }
   total = 0;
   dfsConnection(0);
   if (        ③          )                    //2 分
      printf("connected:yes\n");
   else     printf("connected:no\n");
   memset(visited,0,sizeof(visited));
   int loopFound = 0;
   for(i=0;i<n;++i)
      if (              ④        )             //1 分
         if (dfsLoop(i,-1) )
         {
            loopFound = 1;
            break;
         }
   if (loopFound)     printf("loop:yes\n");
   else       printf("loop:no\n");
   return 0;
}
```