提高组模拟题第五套试题及答案

1．表达式 a×（b＋c）-d / f 转后缀表达式的结果是（　B　）。

A.abc×+ df-/　　B.abc+×　df/-　　C.bc+a×df/-　　D.abc+×-df/

解析：

先算 b+c ，bc+,再算 a*(b+c),abc+*,再算 d/f,abc+*df/,最后 abc+*df/-

2．以下字符串中，字典序最小的是（　A　）。

A.NOIP2017　　B.NOIPC　　C.NOIPC++　　D.NOIPP

解析：

NOIP 都一样，比较 2,C,C，P，ASCII 码种 2 是 50，最小。

3．在 C＋＋中，表达式注意这是大写字母 O（‘O’-‘I’）^13%9＋5 的值是（　D　）。

A.7　　B.8　　C.14　　D.15

解析：

^异或运算符：　0^0=0，0^1=1，1^0=1，1^1=0。位运算运算级别低，先运算 13%9，然后再+5，即 4+5=9，最后 6 异或 9

(79-73)^13%9+5=6^(4+5)=6^9

6　->0110　（异或运算）

9　->1001

15->1111（结果）

结果是 15

4．已知数组 A 中，每个元素 A[I，J] 在存储时要占 3 个字节，设 I 从 1 变化到 8，J 从 1 变化到 10，分配内存时是从地址 SA 开始连续按行存储分配的。试问：A[5，8]的起始地址为(　A　)

A.　SA+141　　B.　SA+180　　C.　SA+222　　D.　SA+225

解析：

数组地址计算问题，只要掌握数据是顺序存储并占用连续的存储空间。注意问题的要求按行存储还是按列存储，就能计算任意单元的起始地址。如题：按行分配空间，则 A[5，8]前 4 行共 40 个单元，第 5 行开始 A[5，1]至 A[5，7]共 7 个单元，即 A[5，8]前有 47 个单元，其地址是

SA+(47*3)=SA+141

5．在 TCP / IP 协议族中，最核心的网络协议是（　　D　）。

A.UDP　　B.HTTP　　C.TCP　　D.IP

解析：

IP 协议是 TCP/IP 协议的核心。

6．应用快速排序的分治思想可以实现一个求第 K 大数的程序。假定不考虑极端的最坏情况，理论上可以实现的最低的算法期望时间复杂度为（　C　）。

A.O(n^2)　　B.O(logn)　　C.O(n)　　D.O(nlogn)

解析：

求第 K 大数，可以在序列中先随机一个数，然后将比这个数小的移到这个数的左边，其余的移到右边，然后可以判断出第 K 大的数在哪一侧，递归处理即可。每次规模期望减少一半，时间复杂度为 T(1)=0(1),T(n)=T(n/2)+n=0(n)。

7．在解决计算机主机与外设之间速度不匹配时通常设置一个缓冲池，主要将计算机输出的数据依次写入该缓冲区，而外设从该缓冲区池中取出数据。该缓冲池应该是一个（　B　）结构。

A.堆栈　　B.队列　　C.二叉树　　D.链表

解析：

主要将计算机输出的数据依次写入该缓冲区，而外设从该缓冲区池中取出数据。需要先进先出，因此是队列的结构。

8.（1E34）16-（676）8 的结果是（　D　）。

A.(1110001111110)2　　B.(7276)10

C.(1C6C)16　　D.(16166)8

解析：1E34 十六进制转十进制：1*16^3+14*16^2+3*16^1+4*16^0=4096+3584+48+4=7732

676 八进制转十进制：6*8^2+7*8^1+6*8^0=384+56+6=446

7732-446=7286

B 不正确，C 1C6C 十六进制转十进制 1*16^3+12*16^2+6*16^1+12=4096+3072+96+12=7276

D 16166 八进制转十进制 1*8^4+6*8^3+1*8^2+6*8^1+6=4096+3072+64+48+6=7286 所以选 D

9．一棵二叉树前序遍历为 ABDECFGH，后序遍历为 EDBGFHCA，以下不是可能的中序遍历的是（　C　）。

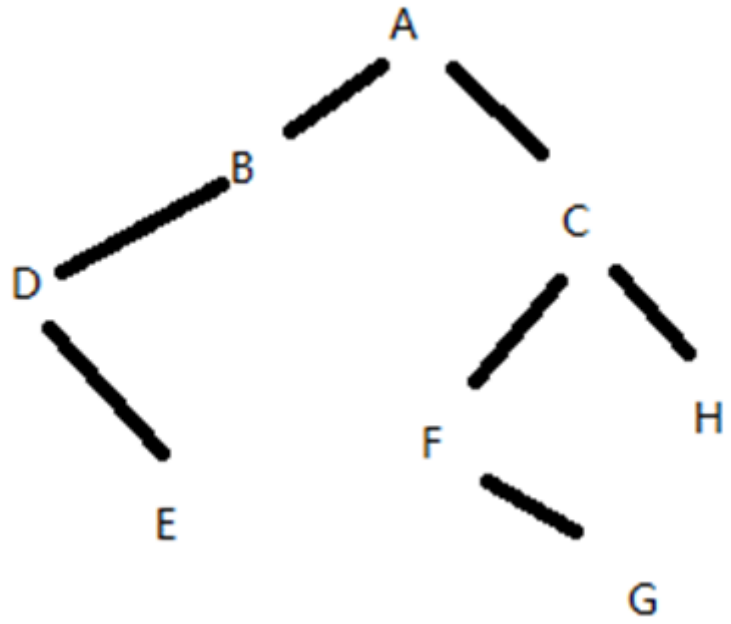
A.DEBAFGCH　　B.EDBAGFCH　　C.DBEAFGCH　　D.BEDAFGCH

解析：A 为根节点

A 选项 DEB 为左子树，FGCH 为右子树

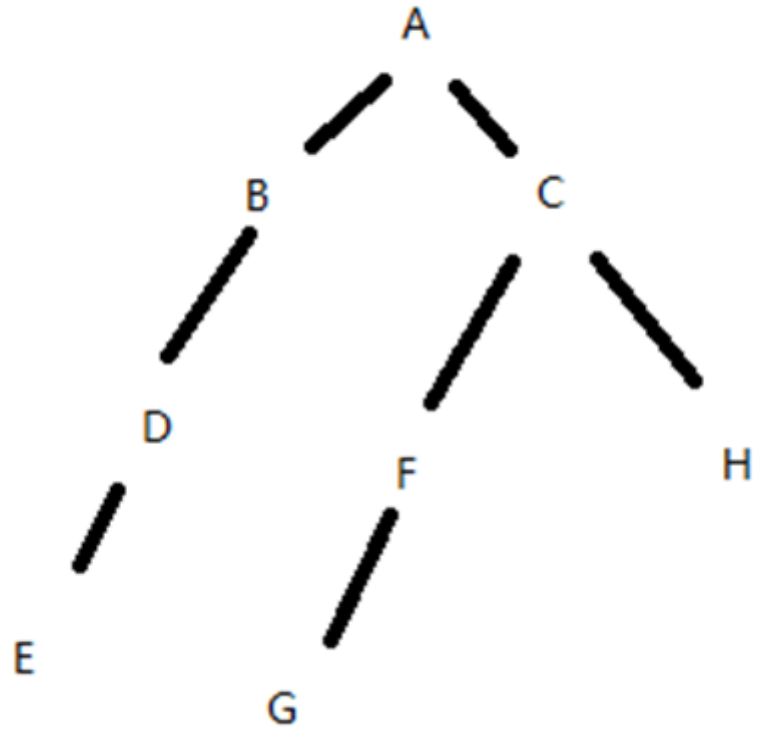
再到先序里面找 BDE，B 为左子树的根节点，DE 是 B 的左子树，D 是根节点 E 是 D 的右子树。

FGCH 是 A 的右子树，C 是根节点 H 是 C 的右子树 F 是 C 的左子树的根节点所以 A 是正确的



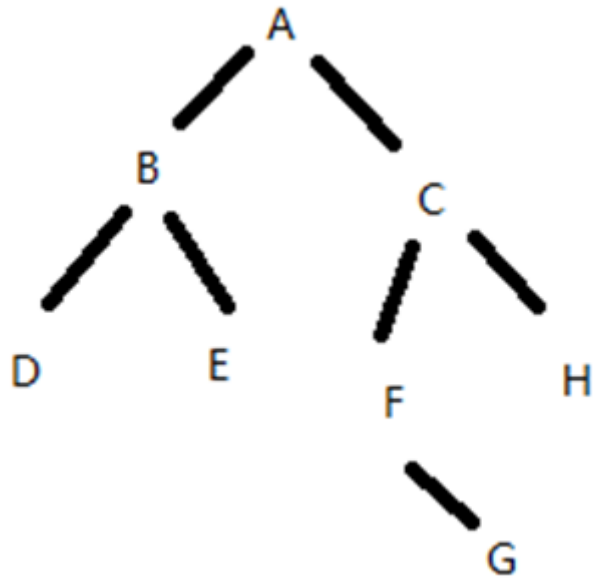
后序遍历 EDBGFHCA

利用 B 选项画出这棵树



后序遍历 EDBGFHCA

C 选项，画出这棵树。



后序遍历：DEBGFHCA，所以 C 是错误的

10. 一个栈的输入顺序为

1,2,3,4,5，下列序列中不可能是栈的输出序列的是（　B　）。

A.54321　　B.24135　　C.32541　　D.13542

解析：

A. 54321 正确

B. 1 入栈 2 入栈 2 出栈 3 入栈 4 入栈 4 出栈，这时候 1 不可能出栈。

11. Linux 是一个用 C 语言写成的开源电脑操作系统内核，有大量的操作系统是基于 Linux 内核创建的。以下操作系统使用的不是 Linux 内核的是（　C　）。

A.Android　　B.CentOS　　C.Windows　　D.Ubuntu

解析：windows 使用的不是 linux 内核。

12. 在下列关于计算机算法的说法中，正确的是（　D　）。

A.对于一个问题，我们能通过优化算法，不断降低其算法复杂度

B.判断一个算法的好坏，主要依据它在某台计算机上具体实现时的运行时间

C.一个算法必须至少有一个输入

D.算法复杂度理论中，P / NP 问题（NP 完全问题）仍是一个未解之谜

解析：

A. 不断降低算法复杂度是错的。

B. 时间复杂度和和空间复杂度。

C. 一个算法有 0 个或多个输入，以刻画运算对象的初始情况，所谓 0 个输入是指算法本身定出了

初始条件；

D．P/NP 是克雷数学研究所公布的 7 个千禧年数学难题之一，目前没有人解出。

13．以下关于二叉树性质中，正确的描述的个数有（　B　）。
A．包含 n 个结点的二叉树的高度至少为 log2n；
B．在任意一棵非空二叉树中，若叶子结点的个数为 n0，度为 2 的结点数为 n2，则 n0=n2+1;
C．深度为 k 的二叉树至多有 2k 个结点
D.没有一棵二叉树的前序遍历序列与后序遍历序列相同
E．具有 n 个结点的完全二叉树的深度为 log2(n+1)
解析：
A．至少为 log2n；错，比如左斜树。
B．正确
C．至多有 2^k-1 个节点，所以错误
D．例如只有一个根结点，前序和后序就相同。
E．具有 n 个结点的完全二叉树的深度为[log2n]+1
14．排序算法是稳定的，这句话的意思是关键码相同的记录排序前后相对位置不发生改变，以下排序算法不稳定的是（　B　）。
A.直接插入排序　　B.快速排序　　C.冒泡排序　　D.归并排序
解析：这个不需要解释了
15．给定 m 种颜色和有 n 个点的手环，要求用这个 m 种颜色给这条手环染色。其中旋转和翻转能够互相得到的算同一种染色方案，如对于 3 个点的手环，ABC 的染色方法与 BCA（旋转）、ACB（翻转）是相同的。当 m=2,n=2 时，一共有三种染色方案，分别为 AA、AB、BB。那么当 m=5,n=4 时，一共有（　D　）种染色方案。
A.625　　B.160　　C.60　　D.120
解析：
所有翻转和旋转组合后，可以得到本题的置换群的置换一共有以下几个：(1,2,3,4)，(2,3,4,1),(3,4,1,2),(4,1,2,3),(1,4,3,2),(2,1,4,3),(3,2,1,4),(4,3,2,1)，分解后的循环数分别为 4,1,2,1,3,2,3,2。根据 Polya 定理，答案为：(2×5^1+3×5^2+2×5^3+5^4)/8 =120。

阅读程序

```
1   # include<iostream>
2   using namespace std;
3
4   unsigned short tot;
5   void Hanoi(int n, char A, char B,char C){
6       ++tot;
7       if(n==1){
8           cout<<A<<"->"<<C<<'/';
9           return;
10      }
11      Hanoi(n-1,A,C,B);
12      cout<<A<<"->"<<C<<'/';
13      Hanoi(n-1,B,A,C);
14  }
15
16  int main(){
17      int n;
18      cin>>n;
19      Hanoi(n,'A','B','C');
20      cout<<'\n'<<tot<<'\n'<<tot<<'\n';
21
22      return 0;
```

```
23  }
```

（1）（1 分）将第 6 行移到 13 行和 14 行之间，输出结果不会变化。（）

答案 ×

解析：注意递归到 n==1 时，如果移动的话++tot 是不会被执行的。

（2）当输人的 n=2 时，输出的第一行为 A->B/B->C/A->C/。（ ）

答案 ×

解析：模拟可得，n==2 时，输出的第一行应当是 A->B/A->C/B->C/。

（3）当输入的 n=3 时，输出的第二行为 8。（ ）

答案 ×

解析：模拟可得，n=3 时，输出的第二行应当是 7。

（4）本程序的含义可以是：有三根柱子，第一根柱子从上到下依次套有编号分别为 1～n 的圆环，现在每次可以移动某个柱子顶部的圆环到另一个柱子的顶部上，并且要求编号较大的圆环要始终不能在编号较小的上面，输出一种操作次数最少的方案以及对应的操作次数。（ ）

解析：答案 √

汉诺塔程序。

（2）

```
1   #include <iostream>
2   #include <iomanip>
3   #include<cstring>
4   using namespace std;
5   const int N=105;
6   int a[N][N];
7   int main(){
8       int n,x,y,count;
9       cin>>n;
10      memset(a,0,sizeof(a));
11      count=a[x=0][y=n-1]=1;
12      while(count<n*n){
13          while(x+1<n&&!a[x+1][y])a[++x][y]=++count;
14          while(y-1>=0&&!a[x][y-1])a[x][--y]=++count;
15          while(x-1>=0&&!a[x-1][y])a[--x][y]=++count;
16          while(y+1<n&&!a[x][y+1])a[x][++y]=++count;
17      }
18      for(x=0;x<n;x++){
19          for(y=0;y<n;y++){
20              cout<<setw(5)<<a[x][y];
21          }
22          cout<<endl;
23      }
24      return 0;
25  }
```

（1）（1 分）删除第 10 行，不影响程序运行结果。（）

答案 √

解析：因为 a 数组已经在全局位置上定义，所以默认为零，因此删除 10 行不影响结果。

（2）将第 12 行改为“while(count<=n*n){”，不影响程序运行结果。（ ）

答案 ×

解析：因为 a[x][y]位置已经填上 1，所以一直是在给下一个位置填数，因此填数到<n*n 即可。填数到<=n*n,程序会进入死循环。

（3）当输人的 n=4 时，程序输出的 a[3][2]的值为 15。（）

答案 ×

解析：输出结果：

10 11 12 1

9 16 13 2

8 15 14 3

7 6 5 4

（4）（3 分）本题的时间复杂度为（ B ）。

A.O(n) B.O(n2) C.O(n3) D.O(n2logn)

解析：

双重循环时间复杂度 O(n2)

（5）当输入的 n＝100 时，a [33][66] 的值为（ D ）。

A.8779 B.8707 C.10957 D.8845

解析：

首先程序所求的应当是从（0，n-1）开始，按照逆时针顺序填数。可以知道第 i 圈一共有 4(n-2i+1)个数。而 a[33][66]所在的恰好是第 34 圈的第一个数，1～33 圈共有(33×(2n-66)/2)×4=8844，因此 a[33][66]=8845

(3)

```
1   #include<iostream>
2   #include <iomanip>
3   #include<cstring>
4   using namespace std;
5   string s;
6   int main(){
7       int k;  //限制输入的 0<=k<26
8       cin>>k>>s;
9       int n=s.length( );
10      for(int i=0;i<n;i++){
11          if(s[i]<='Z'&&s[i]+k>'Z')
12              s[i]=(s[i]+k)%'Z'+'A'-1;
13          else if('A'<=s[i]&&s[i]<='Z')
14              s[i]+=k;
15      }
16      char pre;
17      int st=-1;
```

```
18      for(int i=0;i<n;i++){
19          if(s[i]<'A'||s[i]>'Z'){
20              if(st==-1){
21                  st=i;
22                  pre=s[i];
23              }else{
24                  char tmp=s[i];
25                  s[i]=pre;
26                  pre=tmp;
27              }
28          }
29      }
30      if(st!=-1)
31          s[st]=pre;
32      cout<<s<<endl;
33      return 0;
34  }
```

（1）（1 分）删除第 30 行和第 31 行，不影响程序运行结果。（ ）

答案 ×

解析： s[st] 应当要赋成最后一个非字母的值。

（2）如果输入的 s 不含大写字母，则输出结果与 k 的值无关。（ ）

答案 √

解析：
```
if(s[i]<='Z'&&s[i]+k>'Z')
    s[i]=(s[i]+k)%'Z'+'A'-1;
else if('A'<=s[i]&&s[i]<='Z')
    s[i]+=k;
```
k 值和大写字母有关。

（3）如果知道输出结果，能够反推出唯一的输入结果。（ ）

答案 ×

解析：如果要反推出结果，还需要 k 值。

（4）当 k 的值确定时，不存在两个不同的输入使得输出相同。（ ）

答案 √

解析：当 k 确定时，一个输出是可以唯一确定一个输入的。

（5）如果输入是 6 KU96APY5，则输出为（ B ）。

A.QB96GWE5　　B.QA59GVE6

C.PA59GWF6　　D.PB96GWE5

解析：K+6=Q (U+6)% ‘Z’ =A 所以选 B

（6）（5 分）如果输出是 ab1287F2Tguz，则输入可能为（ C ）。

A.0 ab1287F2Tguz　　B.3 b1287BgTuza

C.16 b12872PgDuza　　D.13 b12872MgAuza

解析：

按照上面方法倒推。

完善程序

1.（拓扑排序）在我们所学的课程中，部分课程之间可能存在依赖关系，如我们在学习图论知识之前，需要先学习离散数学中的基础知识。一门课可能有若干先修课程。现在李老师需要安排一些课程的授课计划，排课需要遵从一定的规则，即只有修习完某课程的全部课程后，才能修习该课程。在本例中，用 1～n 表示 n 个课程，用 x y 表示 x 是 y 的先修课程。输入数据保证图中没有环与重边。要求输出任意一个可行的授课顺序。图用邻接矩阵方法储存。

```
#include <iostream>
#include<cstring>
#include <queue>
using namespace std;
const int N=105;
int a[N][N];
int in[N],s[N];
int n,m,u,v;
void Topo(){
    queue<int> q;
    int cnt;
    for(int i=1;i<=n;i++)
        if(___①___)
            q.push(i);
    while(! q.empty()){
        int cur=q.front();
        q.pop();
        s[cnt++]=___②___;
        for(int i=1;i<=n;i++){
            if(__③___){
            ___④___;
            if(in[i]==0)
                q.push(i);
            }
        }
    }
}
int main(){
    memset(in,0,sizeof(in));
    memset(a,0,sizeof(a));
    cin>>n>>m;
    for(int i=0;i<m;i++){
        cin>>u>>v;
        in[v]++;
        ____⑤___;
    }
    Topo();
    for(int i=0;i<n;i++){
        if(i)
```

```
40              cout<<'';
41          cout<<s[i];
42      }
43      cout<<endl;
44      return 0;
45  }
```

（1）①处应填（ A ）。

A.in[i]==0　　B.in[i]==1　　C.a[1][u]==0　　D.a[1][u]==1

解析：

拓扑排序开始应当将入度为 0 的点加入队列中。

（2）②处应填（ B ）。

A.q.front()　　B.cur　　C.q.back()　　D.s[cur]

解析：

将当前队首的结点插入到当前拓扑序的末尾。

（3）③处应填（ C ）。

A.!--in[i]　　B.a[u][v]==1　　C.a[cur][i]==1　　D.!in[i]

解析：

判断 cur 和 i 之间有没有连边。

（4）④处应填（ B ）。

A.in[i]++　　B.in[i]--　　C.q.pop()　　D.q.push(cur)

解析：

i 的入度减一。

（5）⑤处应填（ C ）。

A.in[u]++　　B.in[u]--　　C.a[u][v]=1　　D.a[v][u]=1

解析：

有向边在邻接矩阵标记为 1，代表有路相连。

2.（8 一皇后问题）要求用回溯法求解 8 一皇后问题，使放置在 8*8 棋盘上的 8 个皇后彼此不受攻击，即：任何两个皇后都不在同一行、同一列或同一斜线上。请输出 8 皇后问题的所有可行解。

```
1   #include<iostream>
2   #include<cmath>
3   using namespace std;
4
5   bool Place(int k, int i,int * x){
6       for(int j=0;j<k;j++)
7           if((x[j]==i)||___ ①___)
8               return false;
9       return true;
10  }
11
12  void NQueens(int k,int n,int * x){
```

```
13      for(int i=0;i<n;i++){
14          if(Place(k,i,x)){
15              _____②____;
16              if(___③__){
17                  for(int j=0;j<n;j++)cout<<x[j]<<" ";
18                  cout<<endl;
19              }
20              else{
21                  ___④___;
22              }
23          }
24      }
25  }
26  int main(){
27      int x[8];
28      for(int i=0;i<8;i++)x[i]=-1;
29      ___⑤_____;
30      return 0;
31  }
```

（1）①处应填（ D ）。

A.x[j]==i-1　　B.abs(x[j]-i)<=1

C.abs(x[j]-k)==i-j　　D.abs(x[j]-i)==k—j

解析：

判断是否在一条斜线上。

（2）②处应填（ C ）。

A.x[i]=k　　B.x[k]=n-i

C.x[k]=i　　D.x[i]=x[k]

解析：

将第 k 行的皇后放在 i 的位置。

（3）③处应填（ B ）。

A.k>n-1　　B.k>=n-1

C.k==n　　D.k>n

解析:

当 k=n-1 时就停止搜索。

（4）④处应填（ B ）。

A.NQueens(k,n-1,x)　　B.NQueens(k+1,n,x)

C.NQueens(k+1,n-1,x)　　D.cout<<x[k]<<" "

解析:

继续搜索下一行。

（5）⑤处应填（ C ）。

A.NQueens(1,8,x)　　B.NQueens(0,7,x+1)

C.NQueens(0,8,x)　　D.NQueens(1,8,x+1）

解析：

从第 0 行开始搜索。