普及组模拟题第四套试题及答案

1、启动计算机引导操作系统是将操作系统(D)

A.从磁盘调入中央处理器 B.从内存储器调入高速缓冲存储器

C.从软盘调入硬盘 D.从系统盘调入内存储器

【解析】操作系统一般存放在系统盘，计算机启动引导系统后，系统中的常用命令就驻留在内存中，方便用户使用计算机。

2、 WINDOWS 9X 是一种( )操作系统(D)。

A.单任务字符方式 B.单任务图形方式

C.多任务字符方式 D.多任务图形方式

【解析】多任务图形方式的操作系统

3．在 24＊24 点阵的字库中，汉字“一”与“编”的字模占用字节数分别是（ A ）。

A.72、72 B.32、32 C.32、72 D.72、32

【解析】首先汉字占用的字节数都是相同的，一个字节可以存储 8 位二进制，24 点就需要 3 个字节，24 行就需要 24*3 即 72 个字节。

4．计算机的运算速度取决于给定的时间内，它的处理器所能处理的数据量。处理器一次能处理的数据量叫字长。已知 64 位的奔腾处理器一次能处理 64 个信息，相当于（ A ）字节。

A．8 个 B．1 个 C．16 个 D．2 个

【解析】一个字节由 8 位二进制组成，64 位奔腾处理器处理 64 个信息，64/8=8 处理 8 字节。

5、算式(2047)10－(3FF)16＋(2000)8 的结果是(A) 。

A)(2048)10 B)(2049)10 C) (3746)8 D) (1AF7)16

【解析】转换成十进制计算 3FF=3*16^2+15*16^1+15*16^0=768+240+15=1023

2000=2*8^3=1024,2047-1023+1024=2048。

6、计算机的运算速度可以用 MIPS 来描述，它的含义是(A)

A．每秒执行百万条指令 B．每秒处理百万个字符

C．每秒执行千万条指令 D．每秒处理千万个字符

【解析】计算机运算速度是指每秒执行指令的条数。M 表示百万，IP 表示指针寄存器，S 表示秒，即每秒执行百万条指令。

7．设栈 S 的初始状态为空，现有 5 个元素组成的序列 {1,2,3,4,5}，对该序列在 S 栈上依次进行如下操作（从序列中的 1 开始，出栈后不再进栈)：进栈、出栈、进栈、进栈、出栈、进栈、出栈、进栈。试问出栈的元素序列是（ D ）。

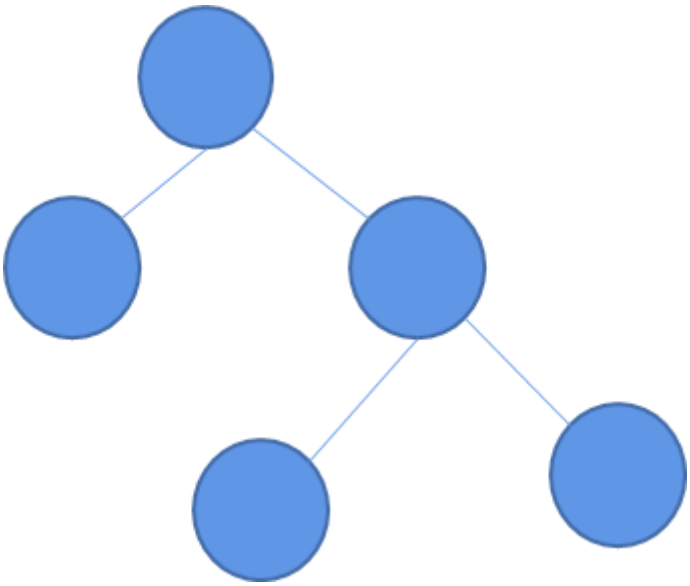
A.{5,4,3,2,1} B.{2,3} C.{2,3,4} D.{1,3,4}

【解析】1 进栈 1 出栈 2 进栈 3 进栈 3 出栈 4 进栈 4 出栈 5 进栈，出栈元素序列就是{1,3,4}

8、在有 n 个子叶节点的哈夫曼树中，其节点总数为( )

A．不确定 B． 2n-1 C． 2n+1 D． 2n

【解析】在哈夫曼树（也叫最优树）中，只有两种类型的结点：度为 0 或 n，即最优二叉树中只有度为 0 或 2 的结点，最优三叉树中只有度为 0 或 3 的结点，所以有 2n-1 个节点 。或者画出一棵哈夫曼树。



9．电线上停着两种鸟（A，B），可以看出两只相邻的鸟就将电线分为了一个线段。这些线段可分为两类：一类是两端的小鸟相同；另一类是两端的小鸟不相同。已知：电线上两个顶点上正好停着相同的小鸟，则两端为不同小鸟的线段数目一定是（　B　）。

A.奇数　　B.偶数　　C.可奇可偶　　D.数目固定

【解析】由于线段两端相同，故此增加一只不同鸟，产生两条两端不同小鸟的线段，增加两只不同鸟，可以产生两条或四条两端不同小鸟的线段。增加 n 只不同小鸟，由于线段两端是相同鸟，通过对称排列，必定是偶数个两端为不同小鸟的线段。

10．从未排序序列中挑选元素，并将其依次放入已排序序列（初始时为空）的一端，这种排序方法称为（　C　）。

A.插入排序　　B.归并排序　　C.选择排序　　D.快速排序

【解析】选择排序，选择排序每次选出最小或者最大的值放入有序序列。注意是从未排序序列中挑选元素。

11．对一个满二叉树，m 个树叶，l 个分支结点，n 个结点，则（　A　）。

A.n=l+m　　B.l+m=2n　　C.m=l-1　　D.n=2l-1

【解析】树叶：度为 0 的结点；分枝结点：度不为 0 的结点；结点：树中的每一个元素都叫结点。所以无论是什么二叉树，叶结点十分枝结点＝结点总数。

12．以下不是操作系统名字的是（　B　）。

A. WindowsXP　　B.Arch/Info　　C.Linux　　D.OS/2

【解析】Arch / Info 是服务程序，不是操作系统。

13．以下不是个人计算机的硬件组成部分的是（　B　）。

A. 主板　　B. 虚拟内存　　C. 总线　　D. 硬盘

【解析】虚拟内存是操作系统软件设置的，不是硬件。

14. 已知元素（8,25,14,87,51,90,6,19,20），这些元素以怎样的顺序全部进入栈中，才能使出栈的顺序满足：8 在 51 前面；90 在 87 的后面；20 在 14 的后面；25 在 6 的前面；19 在 90 的后面？（　D　）

A. 20,6,8,51,90,25,14,19,87　　B. 51,6,19,20,14,8,87,90,25

C. 19,20,90,8,6,25,51,14,87　　D. 6,25,51,8,20,19,90,87,14

【解析】排除法:全部入栈后，8 在 51 的前面出栈，那么 8 就必须排在 51 的后面，A，C 选项不对。

90 在 87 的后面，即 90 排在 87 的前面 B 选项不符合。只剩 D。

看下面表格

1	8 在 51 前面出栈	8 排在 51 的后面	A,C 不符合，
2	90 在 87 的后面出栈	90 排在 87 的前面	B 不符合
3	20 在 14 的后面出栈	20 排在 14 的前面	D 符合
4	25 在 6 的前面出栈	25 排在 6 的后面	D 符合
5	19 在 90 的后面出栈	19 排在 90 的前面	D 符合

15. 假设我们用 d=(a1,a2,...,a5)，表示无向图连通图 G 的 5 个顶点的度数，下面给出的哪组 d 值合理？（　A　）

A. {2,2,2,2,2}　　B. {1,2,2,1,1}　　C. {3,3,3,2,2}　　D. {5,4,3,2,1}

【解析】

（1）无向连通图所有顶点的度之和为偶数。

（2）无向图连通图，有奇数个度的顶点必为偶数个。

阅读程序

(1)

```
1   #include <iostream>
2   #include <cmath>
3   using namespace std;
4
5   bool IsPrime(int num)
6   {
7       for(int i=2;i<=sqrt(num);i++)
8       {
9           if(num%i==0)
10                                  {
11              return false;
12          }
13      }
14      return true;
15  }
16  int main()
17  {
18      int num=0;
19      cin>> num;
20      if (IsPrime(num))
21      {
22          cout<<"YES"<<endl;
23      }
24      else
25      {
26          cout<<"NO"<<endl;
27      }
28      system("pause");
29      return 0;
30  }
```

（1）（1 分）第 19 行输入 97 时，输出为“NO”（不含引号）。（　　）

答案 ×

【解析】本题是判断质数，是输出 YES，不是输出 NO。97 是质数应该输出 YES。

（2）（1 分）第 19 行输入 119 时，输出为“YES”（不含引号）。（　　）

答案 ×

【解析】119 不是质数 17*7=119

（3）若将第 7 行的“＜＝”改成“＜”，程序输出的结果一定不会改变。（　　）

答案×

【解析】例如 49，有因子 7，刚好是 49 的开平方。

（4）当程序执行第 14 行时，i 的值为 sqrt（num）。（　　）

答案×

【解析】因为 i 定义在 for 循环里面只能在 for 循环里面使用，所以第 14 行没有 i 这个变量，就更别提 i 有值了，如果 i 定义在 for 循环上面是 sqrt(num)+1。

（5）（3 分）最坏情况下，此程序的时间复杂度是（　C　）。

A.O(num)　　B.O(nu m²)　　C.O(√num)　　D.O(lognum)

【解析】最坏情况从 2 找到√num。

（6）若输人的 num 为 20 以内的正整数，则输出为“YES”的概率是（　A　）。

A.0.45　　B.0.4　　C.0.5　　D.0.35

【解析】输出 YES 的数有 1 2 3 5 7 11 13 17 19，9 个数，9/20=0.45

这里容易忽略 1，因为程序里面没有判断 if(num<2) return false;

(2)

```
1   #include <bits/stdc++.h>
2   using namespace std;
3   const int mod=2048;
4   long long c,n;
5   long long kasumi(long long x,long long mi){
6       long long res=1;
7       while(mi){
8           if(mi&1){
9               res=(res*x)%mod;
10          }
11          x=(x*x)%mod;
12          mi>>=1;
13      }
14      return res;
15  }
16  int main(){
17      cin>>n>>c;
18      if(n==3){
19          printf("%lld",c*(c-1));
20          return 0;
21      }
22      long long ans=((kasumi(c-1,n)+(c-1)*kasumi(-1,n))%mod+mod)%mod;
23      cout<<ans;
24      return 0;
25  }
```

（1）（1 分）将第 9 行和第 11 行的括号去掉，程序输出结果一定不变。（　　）

答案　√

【解析】res=(res*x)%mod; x=(x*x)%mod;去掉小括号，按照运算优先级运算结果不变。

（2）（1 分）将第 12 行的“mi>>=1”改为“mi*=0.5”，程序输出结果一定不变。（　　）

答案　√

【解析】mi>>=1,mi=mi>>1 相当于 mi/=2;因为 mi 是整型，mi*=0.5 和 mi/=2 结果一样。

（3）若输入为“4 4”，则输出为“78”。（　　）

答案　×

【解析】把 4 4 代入进去结果不是 78，是 84。

（4）此程序的时间复杂度为 O(logn)。（　　）

答案　√

【解析】快速幂的代码，时间复杂度是 O(logn)。因为每次循环都 mi/=2;

（5）若输入为“3 4”，则输出为（ B ）。

A.8 B.12 C.18 D.19

【解析】

```
if(n==3){
19          printf("%lld",c*(c-1));
20          return 0;
21      }
```

此处特判了,n==3 的时候，输出结果是 4*(4-1).，结果是 12。

（6）kasumi（2046,13）的返回值为（ A ）。

A.0 B.2022 C.2 D.2024

【解析】

2046 有因数 2，2^13 是 2048 的倍数，因此答案为 0，或者把数代入进去计算一下。计算次数也不太多。

(3)

```
1   #include <cstdio>
2   int n,r,num[10000];
3   bool mark[10000];
4   void print()
5   {
6       for(int i=1;i<=r;i++)
7       printf("%d ",num[i]);
8       printf("\n");
9   }
10  void search(int x)
11  {
12      for(int i=1;i<=n;i++)
13          if(!mark[i])
14          {
15                          num[x]=i;
16              mark[i]=true;
17              if(x==r) print();
18              search(x+1);
19              mark[i]=false;
20          }
21  }
22  int main()
23  {
24      scanf("%d%d",&n,&r);
25      search(1);
26  }
```

（1）（1 分）程序结束时，对任意 1<=i<=n,mark[i]=0。（ ）

答案 √

【解析】1-n 个数中选出 r 个数的全排列。程序结束时， mark[i]=false; mark 数组的值会回溯。

（2）（2 分）若 n＜r，则程序无输出。（　　）

答案 √

【解析】if(x==r) print();

如果 n<r，这个条件不会执行，没有输出。

（3）（2 分）若输入为“4 3”，则输出中数字 1 和 2 的个数不同。（　　）

答案 ×

【解析】1 和 2 都是 3 个。

（4）（2 分）此程序的时间复杂度为 O(n)。（　　）

答案 ×

【解析】时间复杂度 O(n!)。

（5）若输入为“6 3”，则函数 print 的执行次数为（ B ）。

A.60　　B.120　　C.6　　D.720

【解析】

A_6^3=120。

（6）若输入为“7 4”，则输出的最后一行为（ B ）。

A.4 5 6 7　　B.7 6 5 4　　C.4 3 2 1　　D.1 2 3 4

【解析】

输出的最后一行是字典序最大的排列，7 6 5 4。

完善程序

1. 克鲁斯卡尔求最小生成树思想：首先将 n 个点看作 n 个独立的集合，将所有边快排（从小到大）。然后，按排好的顺序枚举每一条边，判断这条边连接的两个点是否属于一个集合。若不属于同一集合，则将这条边加入最小生成树，并将两个点所在的集合合并为一个集合。若属于同一集合，则跳过。直到找到 n-1 条边为止。

```
1   #include <iostream>
2   #include <algorithm>
3   using namespace std;
4   struct point{
5       int x;
6       int y;
7       int v;
8   };
9   point a[10000];
10
11  int cmp(const point &a,const point &b){
12      if(   ①   ) return 1;
13      else return 0;
14  }
15
```

```
int fat[101];
int father(int x){
    if(fat[x]!=x) return fat[x]=    ②  ;
    else return fat[x];
}

void unionn(int x,int y){
    int fa=father(x);
    int fb=father(y);
    if(fa!=fb) fat[fa]=fb;
}

int main(){
    int i,j,n,m,k=0,ans=0,cnt=0;
    cin>>n;
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        {
            cin>>m;
            if(m!=0){
                k++;
                a[k].x=i;
                a[k].y=j;
                a[k].v=m;
            }
        }
    sort(a+1,a+1+k,  ③  );

    for(i=1;i<=n;i++){
        fat[i]=i;
    }

    for(i=1;i<=k;i++){
        if(father(a[i].x)!=  ④     ){
            ans+=a[i].v;
            unionn(a[i].x,a[i].y);
            cnt++;
        }
        if(   ⑤    ) break;
    }

    cout<<ans;
    return 0;

}
```

（1）①处应填（ A ）。

A.a.v<b.v　　B.a.v>b.v　　C.a.v>=b.v　　D.a.v<=b.v

【解析】

结构体排序，按权值升序排序。

（2）②处应填（　B　）。

A.father(x)　　B.father(fat[x])　　C.fat(father[x])　　D.x

【解析】

if(fat[x]!=x) return fat[x]=②;并查集，找到跟节点，相当于找到祖先（并进行路径压缩处理）。

（3）③处应填（　C　）。

A.algorithm　　B.point　　C.cmp　　D.sizeof(a)

【解析】

调用结构体排序的比较函数（排序规则）。

（4）④处应填（　B　）。

A.a[i].y　　B.father(a[i].y)　　C.fat[a[i].y]　　D.a[i].x

【解析】

查找 father(a[i].y)　的根，如果顶点位置存在且顶点的标记不同，说明不在一个集合中，不会产生回路。

（5）⑤处应填（　D　）。

A.cnt>0　　B.i==1　　C.ans==n-1　　D.cnt==n-1

【解析】

选择了 n-1 条边，求出最小生成树。统计加入集合的边的数量。

子图：从原图中选一些顶点和边组成的图，称为原图的子图。

生成子图：从原图中选所有顶点和一些边组成的图，称为原图的生成子图。

生成树：如果生成子图恰好是一棵树，则称为生成树。

最小生成树：权值之和最小的生成树，则称为最小生成树。

对于 n 个顶点的连通图，只需要 n-1 条边就可以使这个图连通。

2．欧拉回路问题由七桥问题而来，其基本问题是是否能一次性不重复地走遍这七座桥，转换为数学问题中的图论就是指的是从图中的一个顶点出发，是否能够一次性不回头地走遍所有的边，算法代码如下：

```
1   #include <iostream>
2   #include <ctime>
3   using namespace std;
4
5   int G[5][5];
6   int visited[5][5];
7   int n=5;
8   void euler(int u){
9       for(int v=0;v<n;v++){
10          if(G[u][v]&&   ①    ){
11              cout<<u<<"->"<<v<<endl;
12              visited[u][v] = visited[v][u] =   ②   ;
```

```
13              ③
14          }
15      }
16  }
17
18  int main(){
19      G[1][2]=G[2][1]=G[1][3]=    ④    =1;
20      G[2][4]=G[4][2]=G[3][4]=    ⑤    =1;
21      euler(1);
22      return 0;
23  }
```

（1）①处应填（ B ）。

A.G[v][u]　　B.!visited[u][v]　　C.visited[u][v]　　D.visited[v][u]

【解析】

不能访问重复的边，visited 数组标记是否走过，判断没有走过。输出两个顶点，然后标记走过。

（2）②处应填（ A ）。

A.1　　B.0　　C.u　　D.v

【解析】

访问过标记为 1。

（3）③处应填（ A ）。

A.euler(v);　　B.euler(u);　　C.G[u][v]=0;　　D.G[v][u]=0;

【解析】

递归终点 v，从 v 出发继续查找路径。

（4）④处应填（ C ）。

A.G[0][1]　　B.G[1][0]　　C.G[3][1]　　D.G[0][3]

【解析】

G[1][2]=G[2][1]=G[1][3]=G[3][1]=1;初始化邻接矩阵。

（5）⑤处应填（ D ）。

A.G[0][2]　　B.G[2][0]　　C.G[2][1]　　D.G[4][3]

【解析】

G[2][4]=G[4][2]=G[3][4]= G[4][3]=1;

初始化邻接矩阵