普及组模拟题第五套试题及答案

1．文件型病毒传染的主要对象是（　D　）。

A.文本文件　　B.系统文件　　C.可执行文件　　D..EXE 和.COM 文件

【解析】

文件型病毒感染的主要对象是.EXE 和.COM 文件，文件型病毒是对计算机的源文件进行修改，使其成为新的带毒文件。一旦计算机运行该文件就会被感染，从而达到传播的目的。该病毒可以感染基本上所有操作系统的可执行文件。

2.24 针打印机的分辨率约为 180dpi。dpi 数越大，打印精度越高。其中单位 dpi 是指（　C　）。

A.印点／厘米　　B.印点／毫米　　C.印点／英寸　　D.印点／寸

【解析】dpi 是指印点/英寸。dpi 是一个量度单位，用于点阵数码影像，指每一英寸长度中，取样、可显示或输出点的数目。DPI 是打印机、鼠标等设备分辨率的度量单位。是衡量打印机打印精度的主要参数之一，一般来说，DPI 值越高，表明打印机的打印精度越高。

DPI 是指每英寸的像素，也就是扫描精度。DPI 越低，扫描的清晰度越低，web 上使用的图片都是 72dpi，但是冲洗照片不能使用这个参数，必须是 300dpi 或者更高 350dpi。例如要冲洗 4*6 英寸的照片，扫描精度必须是 300dpi，那么文件尺寸应该是(4*300)*(6*300)=1200 像素*1800 像素。

3．内存地址的最重要特点是（　B　）。

A.随机性　　B.唯一性　　C.顺序性　　D.连续性

【解析】注意这里提的是最重要的特点，内存中的每一个基本单位都被赋予一个唯一的序号，称为地址。

4．多媒体计算机是指（　D　）。

A.具有多种功能的计算机　　B.具有多种外设的计算机

C.能处理多种媒体的计算机　　D.能借助多种媒体操作的计算机

【解析】多媒体计算机一般指能够同时接受、处理、存储和展示多种不同类型信息媒体的计算机。

5．最早的计算机的用途是用于（　A　）。

A.科学计算　　B.自动控制　　C.系统仿真　　D.辅助设计

【解析】第一代计算机是电子管计算机，开始于 1946 年，结构上以存储器为中心，使用机器语言，存储量小，主要用于数值计算（科学计算）。多应用于军事方面。

6．CPU 中（　D　）机构相当于运算器中的一个存储单元，它的存取速度比存储器要快得多。

A.存放器　　B.辅存　　C.主存　　D.寄存器

【解析】CPU 主要由运算器、控制器和寄存器组成。寄存器是 CPU 内部的临时存储单元，可以存放数据和地址，也可以存放控制信息和 CPU 工作的状态信息。

7．计算机软件我们一般指的是（　A　）。

A.系统软件和应用软件　　B.应用软件和自由软件

C.培训软件和管理软件　　D.编辑软件和科学计算软件

【解析】软件系统一般都含有很多个软件，这些软件分属于系统软件和应用软件两大类。

8．操作系统在第几代计算机开始应用（　C　）。

A.第一代　　B.第二代　　C.第三代　　D.第四代

【解析】操作系统在第三代计算机开始应用。

9．计算机中的数有浮点与定点两种，其中用浮点表示的数，通常由哪两部分组成（　C　）。

A.指数与基数　　B.尾数与小数　　C.阶码与尾数　　D.整数与小数

【解析】浮点数的表示同数学中的科学计数法有相似之处，由小数及 10 的 N 次幂表示，计算机中的浮点数则将小数部分表示为尾数，将 10 的 N 次幂的 N 作为阶码。

10．如果用一个字节来表示整数，最高位用作符号位，其他位表示数值。例如：00000001 表示＋1,10000001 表示-1，试问这样表示法的整数 A 的范围应该是（　A　）。

A.-127<=A<=127　　B.-128<=A<=128

C.-128<=A<128　　D.-127<A<127

【解析】因为正整数的范围仅能用 7 位的二进制数表示，由于最高位是零，当后 7 位全为 1 时，表示整数 127，再加 1，需要进位，则符号位变为 1，数据发生质的变化，数据由正变为负；而负数道理基本一样。故为-127 至＋127。

11．下列叙述中，正确的是（　D　）。

A.线性表的线性存储结构优于链表存储结构

B.队列的操作方式是先进后出

C.栈的操作方式是先进先出

D.二维数组是指它的每个数据元素为一个线性表的线性表

【解析】排除法：A.线性表的线性存储结构优于链表存储结构，和显然是错误的。

B.队列的操作方式是先进后出,队列是先进先出。

C.栈的操作方式是先进先出，栈是先进后出。

12．用某种排序方法对线性表 25,84,21,47,15,27,68,35,20 进行排序，结点变化如下：

(1)25,84,21,47,15,27,68,35,20;

(2)20,15,21,25,47,27,68,35,84;

(3)15,20,21,25,35,27,47,68,84;

(4)15,20,21,25,27,35,47,68,84;

那么，排序方法是（　D　）。

A.选择排序　　B.希尔排序　　C.合并排序　　D.快速排序

【解析】20,15,21,25,47,27,68,35,84;　先用 25 做基准数，比基准数大的放右边，比基准数小的放左边。继续往下看，也是找基准数，所以是快速排序。

13．如果某二叉树的前序为 STUWV，中序为 UWTVS，那么该二叉树的后序是（　A　）。

A. WUVTS　　B.UWVTS　　C.VWUTS　　D.WUTSV

【解析】由已知二叉树的前序与中序后，可画出二叉树，并得出后序。由于前序为 STUWV，所以根必然是 S，由于中序是 UWTVS，故此二叉树没有右子树。左子树前序为 TUWV 可得左子树根为 T，中序为 UWTV 可得左子树根 T 有一右子根 V，左子树 T 的左子树根为 U，U 有一右子树 W。故此二叉树后序为 A。

14．下面关于数据结构的叙述中，正确的叙述是（　D　）。

A.顺序存储方式的优点是存储密度大，且插入、删除运算效率高

B.链表中的每一个结点都包含一个指针

C.包含 n 个结点的二叉排序树的最大检索长度为 log2n

D.将一棵树转换为二叉树后，根结点没有右子树

【解析】A：顺序存储插入、删除运算效率低；B：链表中的最后一个结点的指针域为空；C：包含 n 个结点的二叉排序树的平均检索长度为（log2n），2 为底 n 的对数。

15．表达式（1＋34）＊5-56／7 的后缀表达式为（　C　）。

A.1 34 + 5 56 7 - * /　　B.- * + 1 34 5 / 56 7

C.1 34 + 5 * 56 7 / -　　D.1 34 5 * + 56 7 /

【解析】

后缀表达式，从左向右开始计算，先计算 1+34，因此 1 34 +再乘以 5，1 34 + 5 *

阅读程序

1．汉诺塔问题：古代有一个梵塔，塔内有三个座 A，B，C，A 座上有 n 个盘子，盘子大小不等，大的在下，小的在上。有一个和尚想把这 n 个盘子从 A 座移到 B 座，但每次只能允许移动一个盘子，并且在移动过程中，3 个座上的盘子始终保持大盘在下，小盘在上，程序如下：

(1)

```
1   #include <iostream>
2   using namespace std;
3   void hanoi(int n,char a,char b,char c){
4       if(n==1)
5           cout<<n<<" "<<a<<" "<<c<<endl;
6       else{
7           hanoi(n-1,a,c,b);
8           cout<<n<<" "<<a<<" "<<c<<endl;
9           hanoi(n-1,b,a,c);
10      }
11  }
12  int main(){
13      int n;
14      cin>>n;
15      hanoi(n,'A','B','C');
16      return 0;
17  }
```

（1）（1 分）当 n>=0 时，程序不会出现死循环。（　　）

答案 ×

【解析】n==0 的时候出现死循环。

（2）（1 分）输出共有 2^n 行。（　　）

答案 ×

【解析】2^n-1 行，因为汉诺塔的最少移动次数就是 2^n-1 次。

（3）当 n>0 时，将第 4 行的“==”改为“＜＝”，程序输出结果必定不变。

答案 √

【解析】当 n 的大于 0 时，n==1 和 n<=1 没什么区别。

（4）将第 5 行的“n”改为“1”，程序输出结果必定不变。（　　）

答案 √

【解析】if(n==1)

　　cout<<n<<" "<<a<<" "<<c<<endl;

n==1 的时候输出，所以输出 1 结果不变。

（5）（3 分）此程序的时间复杂度是（　D　）。

A.O(n)　　B.O(n^2)　　C.O(n^3)　　D.O(2^n)

【解析】递归 n 层，每层 2 个分支。

（6）若要求输出不超过 15 行，则下列哪个 n 的值是合法的？（　B　）

A.0　　B.4　　C.5　　D.6

【解析】 2^n-1 行，因为汉诺塔的最少移动次数就是 2^n-1 次。 所以 n 的值是 4 的时候 2^4-1 的值是 15。所以当 n 的值大于 4 的时候超出 15 行。n 的值是 0 的时候死循环，也不合法。

(2)

```
1   #include <cstdio>
2   #define N 1005
3
4   using namespace std;
5   int num[N];
6
7   int main(){
8       int a1=1,n,x;
9       scanf("%d",&n);
10      num[1]=1;
11      for(int i=1;i<=n;++i){
12          x=0;
13          for(int j=1;j<=a1;++j){
14              num[j]=num[j]*5+x;
15              x=num[j]/10;
16              num[j]%=10;
17          }
18          if(x>0) num[++a1]=x;
19      }
20      printf("0.");
21      for(int i=a1;i<n;++i){
```

```
22          putchar('0');
23      }
24      for (int i=a1;i>=1;i--){
25          printf("%d",num[i]);
26      }
27      putchar('\n');
28      return 0;
29  }
```

（1）（1 分）程序输出的是 5^n 的值。（　　）

答案 ×

【解析】printf(“0.”);根据这条语句，输出的是 0.5 的 n 次方的值。

（2）（1 分）程序执行到第 27 行时，i 的值为 1。（　　）

答案 ×

【解析】

```
24  for (int i=a1;i>=1;i--){
25      printf("%d",num[i]);
26  }
```

i 定义在 for 循环里面，只能在循环里面使用，最后一次循环的时候 i 的值为 0。

（3）对于任意 1<=i<=a1，都有 0<=num[]<=9。（　　）

答案 √

【解析】num[j]%=10; num 数组的值在 0<=num[]<=9 。

（4）程序输出的是一个小数，且小数末尾可能有多余的 0。（　　）

答案 ×

【解析】if(x>0) num[++a1]=x;，x>0 才存入 num 数组，所以末尾没有多余的 0。

（5）此程序的时间复杂度是（　B　）。

A.O(n)　　B.O(n^2)　　C.O(n^3)　　D.O(nlogn)

【解析】双重循环，O(n^2)

（6）若 n=3，则输出为（　B　）

A.8　　B.0.125　　C.0.8　　D.125

【解析】根据前面的程序分析，如果输入的是 3，求的是 0.5 的 3 次方就是 0.125。

3．在起点和终点之间，有 N 块岩石（不含起点和终点的岩石）。在比赛过程中，选手们将从起点出发，每一步跳向相邻的岩石，直至到达终点。

为了提高比赛难度，组委会计划移走一些岩石，使得选手们在比赛过程中的最短跳跃距离尽可能长。由于预算限制，组委会至多从起点和终点之间移走 M 块岩石（不能移走起点和终点的岩石）。

输入文件第一行包含三个正整数 L，N，M，分别表示起点到终点的距离，起点和终点之间的岩石数，以及组委会至多移走的岩石数。

接下来 N 行，每行一个整数，第 i 行的整数 a[i]（0＜a[i]＜L）表示第 i 块岩石与起点的距离。

这些岩石按与起点距离从小到大的顺序给出，且不会有两个岩石出现在同一个位置。

输出文件只包含一个整数，即最短跳跃距离的最大值。

```
1   #include <iostream>
2   using namespace std;
3   int l,n,m,a[50005],ans;
4   bool check(int dis)
5   {
6       int count=0,last=0;
7       for(int i=1;i<=n;i++)
8           if(a[i]-last<dis) count++;
9           else last=a[i];
10      if(count>m) return 0;return 1;
11  }
12  int main()
13  {
14      ios::sync_with_stdio(0);
15      cin>>l>>n>>m;
16      for(int i=1;i<=n;i++)
17          cin>>a[i];
18      a[n+1]=1;
19      int fl=0,fr=l;
20      while(fl<=fr)
21      {
22          int mid=(fl+fr)/2;
23          if(check(mid)) fl=mid+1,ans=mid;
24          else fr=mid-1;
25      }
26      cout<<ans;
27      return 0;
28  }
```

（1）（1 分）将第 19 行的“fl=0’”改为“fl=1”，程序输出结果必定不变。（　　）

答案 √

【解析】ans 初始值为 0，所以 0 的问题解决了，二分结果不变。

（2）（2 分）程序执行到第 26 行时，必有 fl>fr。（　　）

答案 √

【解析】while(fl<=fr) 执行到 26 行，fl>fr。

（3）（2 分）若第 23 行执行的 check(mid)==1，则最终的 ans 小于或等于此时的 mid。（　　）

答案 ×

【解析】因为 check 函数的返回值不是 0，就是 1，所以改过之后值不发生改变。

（4）此程序的时间复杂度是（　C　）。

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

【解析】check 是 O(n)，二分是 O(logl)。

（5）若输入为：

25 5 2

2

11

14

17

21

则输出为（ B ）。

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

解析：代入程序模拟，移走 2 和 14。

完善程序

1．迪杰斯特拉算法是由荷兰计算机科学家狄克斯特拉于 1959 年提出的，因此又叫狄克斯特拉算法。是从一个顶点到其余各顶点的最短路径算法，解决的是有向图中最短路径问题。迪杰斯特拉算法主要特点是以起始点为中心向外层层扩展，直到扩展到终点为止。

```
#include <iostream>
using namespace std;
int main()
{
    int edgs;
    int points;
    int dis[10];
    int flag[10];
    int infinity=9999999;
    cin>>points>>edgs;
    int edg[10][10];
    for (int i=1;i<=points;i++)//初始化
    {
        for(int j=1;j<=points;j++)
        {
            if(i==j)
            {
                edg[i][j]=    ①    ;
            }
            else
            {
                edg[i][j]=    ②    ;
            }
        }
    }
    int point1,point2,quanzhi;
    for(i=1;i<=edgs;i++)
    {
        cin>>point1>>point2>>quanzhi;
        edg[point1][point2]=  ③      ;
    }
    for(i=1;i<=points;i++)
    {
```

```
34          dis[i]=edg[1][i];
35      }
36      for(i=1;i<=points;i++)
37      {
38          flag[i]=0;
39      }
40      flag[1]=1;
41      int min,u;
42      for(i=1;i<=points-1;i++)
43      {//源点到源点不用比较，因次总的次数少一次
44          min=infinity;
45          for(int j=1;j<points;j++)
46          {
47              if(flag[j]==0&&dis[j]<min)
48              {//核心思想：依次比较出离源点最近的点
49                  min=     ④     ;
50                  u=j;
51              }
52          }
53          flag[u]=1;
54          for(int v=1;v<=points;v++)
55          {//找出离源点最近的点后更新 dis 里面的源点到各个点的值是否最小
56              if(edg[u][v]<infinity)
57              {
58                  if(dis[v]>dis[u]+edg[u][v])
59                  {
60                      dis[v]=    ⑤     ;
61                  }
62              }
63          }
64      }
65      for(i=1;i<=points;i++)
66      {
67          cout<<dis[i]<<" ";
68      }
69      cout<<endl;
70  }
```

（1）①处应填（ C ）。

A.infinity B.dis[j] C.0 D.1

【解析】i==j，自己到自己的最短距离初始化为 0。

（2）②处应填（ A ）。

A.infinity B.dis[j] C.0 D.1

【解析】int infinity=9999999;，邻接矩阵初始化为最大值。

（3）③处应填（ A ）。

A.quanzhi B.0 C.inf D.1

【解析】把边权赋值给邻接矩阵。

（4）④处应填（ B ）。

A.j　　B.dis[j]　　C.flag[j]　　D.i

【解析】if(flag[j]==0&&dis[j]<min) 根据前面这条语句，应选择 dis[j]更新最短距离。

（5）⑤处应填（　　C　　）。

A.dis[u]　　B.edg[u][v]　　C.dis[u]+edg[u][v]　　D.infinity

【解析】

```
if(dis[v]>dis[u]+edg[u][v])
59{
60   dis[v]=      ⑤     ;
61}
```

dis[u]+edg[u][v]<dis[v],就更新 dis[v]= dis[u]+edg[u][v];

2．完全背包问题。容量为 10 的背包，有 5 种物品，每种物品数量无限，其重量分别为 5,4,3,2,1，其价值分别为 1,2,3,4,5。设计算法，实现背包内物品价值最大。代码如下（输出 50):

```
1   #include <iostream>
2   #include <algorithm>
3
4   using namespace std;
5
6   int main()
7   {
8       int total_weight=10;
9       int w[6]={0,5,4,3,2,1};
10      int v[6]={0,1,2,3,4,5};
11      int dp[11]={     ①     };
12
13      for(int i=1;i<=   ②      ;i++)
14          for(int j=w[i];j<=    ③    ;j++)
15              dp[j]=     ④       ;
16
17      cout<<    ⑤     <<endl;
18      return 0;
19  }
```

（1）①处应填（　A　）。

A.0　　B.5　　C.10　　D.15

【解析】dp 数组初始化为 0。

（2）②处应填（ A　　）。

A.5　　B.6　　C.10　　D.15

解析：i 枚举的物品种类，共有 5 种物品。

（3）③处应填（　C　）。

A.5　　B.6　　C.10　　D.15

【解析】j 循环背包容量，背包容量为 10。

（4）④处应填（　D　）。

A.dp[j]+v[i]

B.dp[j-w[i]]+v[i]

C.min(dp[j],dp[j-w[i]]+v[i])

D.max(dp[j],dp[j-w[i]]+v[i])

【解析】取最大价值。我们要决定是不是要放这个物品，就从这个物品的大小出发遍历背包容量，然后每次遍历都对比下假如现在腾出这个物品的空间并且放进去比原来的价值还大的话，就放进去。

（5）⑤处应填（　B　）。

A.v[10]　B.dp[10]　C.w[10]　D.total_weight

【解析】压掉数组的 i 这一维，因此背包容量为 10，结果存储在 dp[10]里面。