普及组模拟题第九套试题及答案

1．关于CPU下面哪些说法是正确的？（　B　）

A.CPU全称为中央控制器

B.CPU能直接运行机器语言

C.CPU最早是由Intel公司发明的

D.同样主题下，32位的CPU比16位的CPU运行速度快一倍

【解析】

CPU，即central processing unit，中央处理器；CPU只能执行机器指令，也就是二进制的代码位数只能说明处理的字长，所在的系统硬件指令不同，速度很难说谁快；Intel最早发明的是微处理器，而CPU之前就由电子管、晶体管实现。

2．在字长为16位的系统环境下，一个16位带附号整数的进制补码为1111111111101101。其对应的十进制整数应该是（　B　）

A.19　　B.-19　　C.18　　D.-18

【解析】

负数的补码，首先是除符号位外，其他取反码，然后负数的补码+1，反过来计算即可。

负数补码加过1，所以1111111111101100，再转换（反码）1000000000010011.转十进制

1*2^4+1*2^1+1*2^0=16+2+1=19,即-19

3．在计算机内部，用来传送、存储、加工处理的数据或指令（命令）都是以（　B　）形式进行的。

A.十进制码　　B.二进制码　　C.智能拼音码　　D.五笔字型码

解析：常识题 。

4．排序算法是稳定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，下列哪种排序算法不是稳定的？（ D　）

A.插入排序　　B.基数排序　　C.归并排序　　D.堆排序

【解析】稳定排序：冒泡排序，插入法排序，归并排序，桶排序，计数排序，基数排序

不稳定排序：选择排序，快速排序，堆排序，希尔排序

5．一棵6节点二叉树的中序遍历为DBAGECF，先序遍历为ABDCEGF，后序遍历为（　C　）

A.DCBEFAC　　B.CBEACFD　　C.DBGEFCA　　D.ABCDEFG

【解析】先看先序遍历A是根节点，后序遍历应该最后一个字符是A，所以只有C。

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时间复杂度为（　B　）

A.O(logn)　　B.O(nlogn)　　C.O(n)　　D.O(1)

【解析】快速排序极端的最坏情况，时间算法复杂度是O(n^2)。非极端情况下，平局时间复杂度和最快时间复杂度都是O(nlogn)。

7．若 3 个顶点的无权图 G 的邻接矩阵用数组存储为 {{0,1,1}，{1,0,1}，{0,1,0}}，假定在具体存储中顶点依次为：V1,V2,V3。关于该图，下面的说法哪个是错误的？（　　C　　）

A.该图是有向图　　B.该图是强联通的

C.该图所有顶点的入度之和减所有顶点的出度之和等于 1

D.从 v1 开始的深度优先遍历所经过的顶点序列与广度优先的顶点序列是相同的

【解析】

0,1,1

1,0,1

0,1,0

v1~v3 右边，v3~v1 无边，所以 A 正确，有向图。B 正确，是强连通图，D 正确都是 1-2-3

C 入度之和和出度之和相等。所以答案是 C

8. 2019 年 10 月 14 日是星期一，1978 年 10 月 14 日是（　D　　）。

A.星期日　　B.星期五　　C.星期一　　D.星期六

【解析】

基姆拉尔森计算公式计算：

(d+2*m+3*(m+1)/5+y+y/4-y/100+y/400+1)%7=(14+2*10+3*(10+1)/5+1978+1978/4-1978/100+1978/400+1)%7=(14+20+6+1978+494-19+4+1)=2498%7=6

9．表达式 a*(b+c)-d 的后缀表达式是（　B　　）。

A.abcd*+-　　B.abc+*d-　　C.abc*+d-　　D.-+*abcd

【解析】

先算 b+c,bc+,然后*a，abc+*所以选 B

10．某算法计算时间表示为递推关系式：T(N)=N+T(N/2)，则该算法时间复杂度为(　B　　)。

A.O(N2)　　B.O(NlogN)　　C.O(N)　　D.O(1)

解析：答案 B

11．如果根结点的深度记为 1，则一棵恰有 2011 个叶子结点的二叉树的深度不可能是

(　　A　　)。

A.11　　B.12　　C.13　　D.2011

【解析】满二叉树的深度与叶子结点的个数关系 2^(k-1)，深度为 11 时，叶子结点个数是 2^10，即为 1024 个，所以不可能有 2011 个叶子节点。

12．对于序列“7,5,1,9,3,6,8,4”在不改变顺序的情况下，去掉（　D　　）会使逆序对的个数减少 3。

A.7　　B.5　　C.4　　D.6

【解析】从后往前找，去掉 4，减少 7 4 ，5 4， 9 4， 6 4， 8 4 这些逆序对

去掉 8，减少 9 8，8 4。去掉 6，减少 6 4 ，7 6，9 6 三组，所以选择 D

13．某班有 50 名学生，每位学生发一张调查卡，上写 a、b、c 三本书的书名，将读过的书打✓，结果统计数字如下：只读 a 者 8 人；只读 b 者 4 人；只读 c 者 3 人；全部读过的有 2 人；读过 a，b 两本书的有 4 人；读过 a、c 两本书的有 2 人；读过 b、c 两本书的有 3 人。则读过 a 的人数是（　C　）。

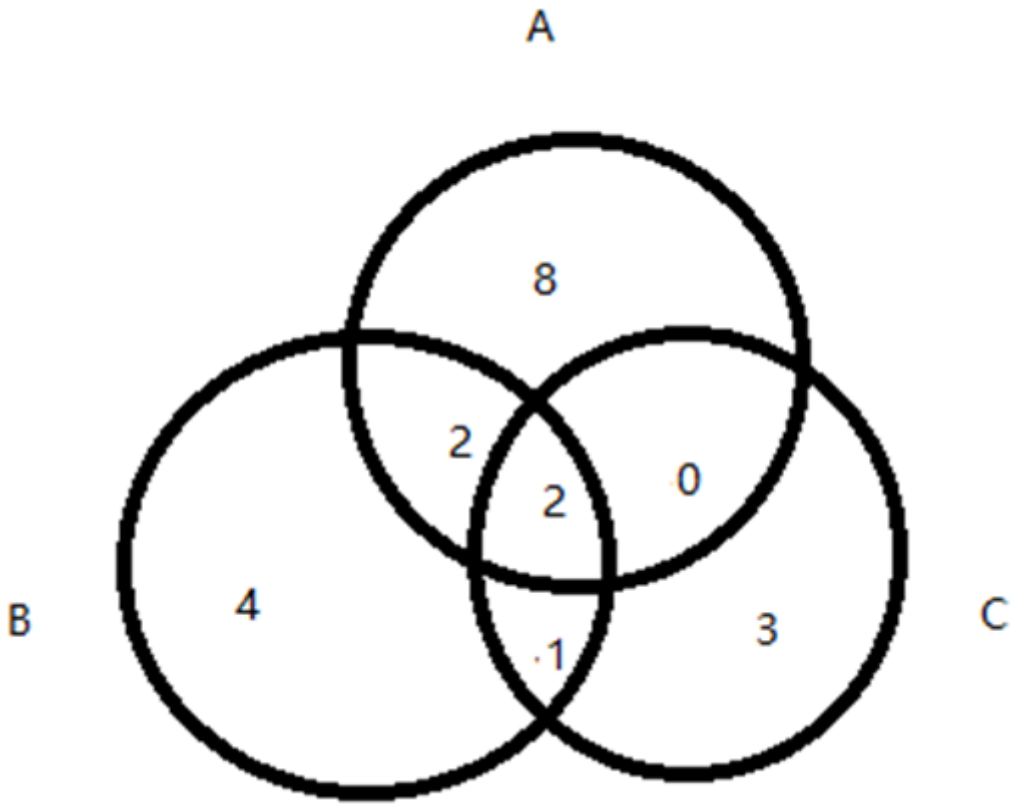
A.10 人　　B.30 人　　C.12 人　　D.24 人

解析：

r(a)=8，r(b)=4，r(c)=3，r(abc)=2

r(ab)=4，r(ac)=2，r(bc)=3

画出右边图形，可知读过 a 的有 12 人



14．一家三口人，恰有两个人生日在同一天的概率是（　C　　）。（假设每年都是 365 天）

A.1/365　　B.365/ (364 X 365)　　C.(3X364)/(365X365)　　D.1/12

【解析】

$$\frac{C_3^2 \times A_{365}^2}{365 \times 365 \times 365}$$

$$\frac{3 \times 364}{365 \times 365}$$

15．字符串“abeab”本质不同的子串（不包含空串）个数（　D　　）

A. 15　　B. 14　　C. 13　　D. 12

【解析】

子串的定义是原字符串中连续的一段字符组成的字符串，不同的子串是当且仅当两个子串长度不一样，或者长度一样但有至少任意一位不一样时成立。那么将“abeab”先看成 5 个不同字符，不同字符的子串个数是 n*(n+1)/2=15,不包含空串。字串只有一个字符时，多了一个“a”，一个“b”；字串只有 2 个字符是多了一个“ab”。15-3=12。

前面枚举过类似的字符串。

阅读程序

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 int main(){
4   int a[4], b[4];
5   int i,j,tmp;
6   for(i=0;i<4;i++)
7           scanf("% d",&b[i]);
8   for(i=0; i<4; i++){
9       a[i]=0;
10        for(j=0; j<=i; j++){
11            a[i]+=b[j];
12            b[a[i]%4]+=a[j];
13          }
14   }
15  tmp=1;
16    for(i=0;i<4;i++){
17           a[i] %=10;
18           b[i] %=10;
19          tmp *=a[i]+b[i];
20   }
21  printf(" % d\n",tmp);
22  return 0;
23   }
```

（1）若输入的 b 数组都是奇数，程序运行到第 15 行时，b[2]没有发生变化。（　　）

答案 ×

【解析】若输入的 b 数组都是奇数，程序运行到第 15 行时，b[1]没有发生变化

（2）若将第 11 行和第 12 行交换，程序的结果可能发生变化。（　　）

答案 √

【解析】运算的先后顺序改变，可能会影响程序的结果。

（3）当程序运行到第 15 行时，此时 a 数组的值等于 b 数组的值。（　　）

答案 ×

若输入 2357，当程序运行到第 15 行时，a 数组的值不等于 b 数组的值。

（4）若输入 b 数组都是偶数，当程序运行到第 15 行时，a 数组中的值也全部为偶数。（　　）

答案 √

【解析】输入 b 数组都是偶数，经过第 11 行和第 12 行后，a 数组中的值也全部为偶数。

（5）该程序能输出的最大结果为（　C　）。

A.6561　　B.10000　　C.104976　　D.43046721

【解析】

该程序能输出的最大结果为 18x18x18x18=104976。

（6）若输入 2 3 5 7，输出的结果为（　B　）。

A.210　　B.5850　　C.3360　　D.103680

【解析】模拟程序执行过程，b={94,8,49,33},a={2,5,26,92}，结果为 5850。

(2)

```
1   #include <ctype.h>
2   #include <stdio.h>
3   void expand(char s1[],char s2[])
4  {
5        int i,j,a,b,c;
6        j=0;
7        for(i=0;(c=s1[1])!='\0';i++)
8           if(c=='-')
9           {
10            a=s1[i-1]; b=s1[i+1];
11            if(isalpha(a) && isalpha(b) || isdigit(a) && isdigit(b))
12            {
13              j--;
14             do
15               s2[j++]=a++;
16             while(tolower(a) < tolower(s1[i+1]));
17            }
18         else s2[j++]=c;
19       }else s2[j++]=c;
20       s2[j]='\0';
21   }
22  int main()
23  {
24     char s1[100],s2[300];
25     printf("input s1:");
26     gets(s1);
27     expand(s1,s2);
28     printf("%s\n",s2);
29}
```

函数 isalpha（a）用于判断字符 a 是否为字母，isdigit（b）用于判断字符 b 是否为数字，如果是，返回 1，否则返回 0。

函数 tlower（a）的功能是当字符 a 是大写字母，返回其小写字母，其余情况不变

规定：输入的字符串只包含大小写字母，数字和“一”，且保证“一”不会出现在首位和末位。

（1）若输入的字符串不包含“一”号，则输出的字符串和输入相同。（）

答案　√

【解析】不包含“一”号，s2 和 s1 必定相同。

（2）若输入的字符串包含“一”号，输出的字符串长度可能与输入的相同。（）

答案 √

【解析】输入 a1-a2，输出的字符串为 a1-a2，则输出的字符串长度与输入的相同。

（3）若存在一个字符，只存在于 s2 字符串中而不存在于 s1 字符串中，则 s1 字符串中一定存在"一"号。（）

答案 √

【解析】if((c=='-')才会处理字符串，若输人的字符串不包含“一”号，则输出的字符串和输入相同。

（4）若存在一个字符，只存在于 s2 字符串中而不存在于 s1 字符串中，则该字符一定不可能是大写字母。（ ）

答案 ×

【解析】如果输入 A-D，输出结果是 ABCD。

（5）若输入 6 个字符，最多输出（ C ）个字符。

A.6 B.20 C.52 D.54

解析：

A-Z-A-Z，输出 26×2=52 个字符。

（6）若输出 500 个字符，至少输入（ C ）个字符。

A.20 B.5 C.60 D.500

解析：

A~Z 一组可以输出 26 个字符，至少输入 60 个字符。

(3)

```
#include<iostream>
# include<cstring>
#include<cstdio>
 #define N 500+10
 using namespace std;
 int a[N],n;
 int main()
 {
     cin>>n;
     for(int i=1;i<=n;++i)cin>>a[i];
     for(int i=1;i<=n;++i)
     {
        int tmp=i;
        for(int j=i+1;j<=n;++j)
          if(a[j]<a[tmp])    tmp=j;
        swap(a[i],a[tmp]);
      }
    for(int i=1;i<=n;++i)cout<<a[i]<<' ';
    cout<<endl;
      return 0;
```

```
21    }
```

（1）上述代码实现了对一个长度为 n 的序列进行排队。（）

答案 √

【解析】正确，选择排序，实现了对一个长度为 n 的序列排序（由小到大）。

（2）去掉头文件“＃include<cstdio>”后程序仍然正常编译运行。（）

答案 √

【解析】正确，没用到 scanf 和 printf 所以去掉＃include<cstdio>程序可以正常编译。

（3）去掉“using namespace std; ”后程序仍然正常编译运行。（ ）

答案 ×

【解析】错误，如果去掉“using namespace std; ”，cin，cout 前面需要加上 std::。

（4）我们将上述算法称为（ C ）

A.KMP　　B.冒泡排序　　C.选择排序　　D.归并排序

【解析】

每一个位置选择当前最小的元素。

（5）上述代码的时间复杂度为（ C ）

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

【解析】

选择排序，双重循环，时间复杂度是 O(n^2)。

（6）若输入数据为：

5

3 2 1 5 4

则 if（a [j] <a [tmp]）这句话中的条件会成立（ C ）次（即后面的 tmp=j 会被执行多少次）

A.5　　B.7　　C.3　　D.4

解析：交换 3 次。

3 2 1 5 4

1 2 3 5 4

交换两次

```
for(int i=1;i<=n;++i)
 {
        int tmp=i;
        for(int j=i+1;j<=n;++j)
            if(a[j]<a[tmp]) tmp=j;
        swap(a[i],a[tmp]);
 }
```

执行 j 循环

第一次交换 a[2]<a[1] tmp=2;

第二次交换 a[3]<a[2] tmp=3;

最后 swap(a[1],a[3]);

数列变成 1 2 3 5 4

最后 5 4 交换一次，共计三次。

完善程序

1.（最大连续子段和）给出一个数列（元素个数不多于 100），数列元素均为负整数、正整数、0。请找出数列中的一个连续子数列，使得这个子数列中包含的所有元素之和最大，在和最大的前提下还要求该子数列包含的元素个数最多，并输出这个最大和以及该连续子数列中元素的个数。例如数列为 4，-5,3,2,4 时，输出 9 和 3；数列为 1,2,3，-5,0,7,8 时，输出 16 和 7。

```
1 #include <iostream>
2 using namespace std;
3 int a[101];
4 int n,i,ans,len, tmp, beg,end;
5 int main(){
6       cin>>n;
7       for(i=1;i<=n;i++)
8            cin>>a[i];
9       tmp=0;
10      ans=0;
11      len=0;
12      beg=_____①________;
13      for(i=1;i<=n;i++){
14           if(tmp+a[i]>ans){
15                ans=tmp+a[i];
16                len=i-beg;
17           }
18          else if(    ___②___    &&i-beg>len)
19                len=i-beg;
20          }
21          if(tmp+a[i]_____ ③_____ ){
22              beg=_____④___;
23             tmp=0;
24          }else
25              ____⑤____;
26     }
27    cout<<ans<<" "<<len<<endl;
28    return 0;
29  }
```

（1）①处应填（ A ）。

A.0　B.1　C.n　D.-1

【解析】beg 存储的是当前最优数列的首项的前一项序号，初始化为 0。

（2）②处应填（ C ）。

A.temp==ans　B.a[i]+tmp<=ans　C.a[i]+tmp==ans　D.tmp<=ans

【解析】

如果 a[i]+tmp 与 ans 相等时，和最大的前提下还要求该子数列包含的元素个数最多，当 i-beg 大于 len，就会更新 len=i-beg。

（3）③处应填（ C ）。

A.==0　B.<ans　C.<0　D.<=0

解析

当 tmp 要清零时，i 位置的元素不会是最优数列的元素，此时 a[i]+tmp<0。

（4）④处应填（　C　）。

A.1　　B.n　　C.i　　D.0

解析

当 a[i]+tmp<0，i 位置的元素不会是最优数列的元素，beg 会更新到 i 位置。

（5）⑤处应填（　D　）。

A.tmp=0　　B.len++　　C.beg=i　　D.tmp+=a[i]

解析

a[i]+tmp>=0 时，a[i]加入 tmp 中。

2.(棋盘覆盖问题）在一个 2^k×2^k 个方格组成的棋盘中恰有一个方格与其他方格不同（如图中标记为-1 的方格），称之为特殊方格。现用 L 形（占 3 个小方格）纸片覆盖棋盘上除特殊方格的所有部分，各纸片不得重叠，于是，用到的纸片数恰好是（4^k-1)/3。在下面给出的覆盖方案例子中，k=2，相同的 3 个数字构成一个纸片。下面给出的程序使用分治法设计的，将棋盘一分为四，依次处理左上角、右上角、左下角、右下角，递归进行。

请将程序补充完整。例：

2	2	3	3
2	-1	1	3
4	1	1	5
4	4	5	5

-1 为特殊方格的位置，其他数字表示 L 形方格放置的顺序

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 int board [65] [65], tile; / / tile 为纸片编号
4 void chessboard( int tr, int tc, int dr, int dc, int size)
5 { / *dr，dc 依次为特殊方格的行、列号* /
6       int t,s;
7       if (size == 1)
8           return;
9       t=tile++;
10      s=size/2;
11      if(  ___①___    )
12          chessboard(tr,tc,dr,dc,s);
13      else{
14          board[tr+s-1][tc+s-1]=t;
15          ____②__ ;
16      }
```

```
17     if(dr <tr +s&&dc<tc+s)
18        chessboard(tr,tc+s,dr,dc,s);
19     else{
20        board[tr+s-1][tc+s]=t;
21        ____③___;
22      }
23      if(dr>=tr+s&& dc<tc+s)
24         chessboard(tr+s,tc,dr,dc,s);
25      else{
26         board[tr+s][tc+s-1]=t;
27          ____④_____;
28     }
29     if(dr>=tr+s&&dc>=tc+s)
30         chessboard(tr+s,tc+s,dr,dc,s);
31     else{     board[tr+s][tc+s]=t;
32          ___⑤_____;}
33      }
34
35   void prtl(int b[][65],int n)
36   {
37        int i,j;
38        for(i=1;i<=n;i++)
39        {
40          for(j=1;j<=n;j++)
41              cout<<setw(3)<<b[i][j];
42          cout<<endl;
43         }
44    }
45
46    int main(){
47       int size,dr,dc;
48    cout<<"input size(4/8/16/64):"<<en <endl;
49       cin>> size;
50       cout<<"input the position of special block(x,y):"<<endl;
51       cin>>dr>>dc;
52       board[dr][dc]=-1;
53       tile++;
54       chessboard(1,1,dr,dc,size);
55   prtl( board,size);
56   }
```

（1）①处应填（　C　）。

A.(dr<tr+s)&&(dc>=tc+s)

B.(dr<=tr+s)&&(dc<=tc+s)

C.(dr<tr+s)&&(dc<tc+s)

D.(dr>=tr+s)&&(dc<tc+s)

解析

将棋盘分成了四等份，如果特殊方格（dr，dc）落在左上角的子棋盘中，即 (dr<tr+s)&&(dc<tc+s)则需要进一步递归，（tr，tc）和（dr，dc）不变并且只缩小棋盘 s=s/2。

（2）②处应填（　B　）。

A.chessboard(tr+s,tc,t+s,tc+s-1,s)

B.chessboard(tr,tc,tr+s-1,tc+s-1,s)

C.chessboard(tr,tc+s,tr+s-1,tc+s,s)

D.chessboard(tr,tc,tr+s,tc+s,s)

解析

如果特殊方格落（dr，dc）不落在左上角的子棋盘中，用编号为 t 的骨牌覆盖子棋盘右下角，board[tr+s-1][tc+s-1]=t，并进行覆盖其余方格 chessboard(tr,tt,tr+s-1,tc+s-1)。

（3）③处应填（　D　）。

A.chessboard(tr,tc,tr+s-1,tc+s-1,s)

B.chessboard(tr+s,tc,tr+s,tc+s-1,s)

C.chessboard(tr+s,tc+s,tr+s,tc+s,s)

D.chessboard(tr,tc+s,tr+s-1,tc+s,s)

解析

如果特殊方格（dr，dc）不落在右上角的子棋盘中，用编号为 t 的骨牌覆盖子棋盘左下角，board[tr+s-1][tc+s]=t，并进行覆盖其余方格 chessboard (tr,tc+s,tr+s-1,tc+s,s)。

（4）④处应填（　D　）。

A.chessboard(tr,tc+s,tr+s-1,tc+s,s)

B.chessboard(tr+s,tc+s,tr+s,tc+s,s)

C.chessboard(tr,tc,tr+s-1,tc+s-1,s)

D.chessboard(tr+s,tc,tr+s,tc+s-1,s)

解析

如果特殊方格（dr，dc）不落在左下角的子棋盘中，用编号为 t 的骨牌覆盖子棋盘右上角，broad[tr+s][tc+s-1]=t 并进行覆盖其余方格 chessboarrd(tr+s,tc,tr+s,tc+s-1,s)。

（5）⑤处应填（　D　）。

A.chessboard(tr,tc,tr+s-1,tc+s-1,s)

B.chessboard(tr,tc+s,tr+s-1,tc+s,s)

C.chessboard(tr+s,tc,tr+s,tc+s-1,s)

D.chessboard(tr+s,tc+s,tr+s,tc+s,s)

解析

如果特殊方格（dr，dc）不落在右下角的子棋盘中，用编号为 t 的骨牌覆盖子棋盘左上角，board[tr+s][tc+s]=t，并进行覆盖其余方格 chessboard(tr+s,tc+s,tr+s,tc+s,s)。