提高级 CSP-S 第 2 套初赛模拟试题

一、单项选择题(共 15 题,每题 2 分,共计 30 分;每题有且仅有一个正确选项)

1. 今有一空栈 S,对下列待进栈的数据元素序列 1,2,3,4,5,6,7,8,9…进行以进栈、进栈、出栈、进栈、进栈、进栈、出栈为一次操作的规律一直进行操作,那么在 2019 次操作后,S 栈的栈顶元素为(　　)。

A. 10090　　B. 10094　　C. 10100　　D. 10086

2. 对有序数组{5,13,19,21,37,56,64,75,88,92,100}进行二分查找,等概率的情况下查找成功的平均查找长度(平均比较次数)是(　　)。

A. $\frac{35}{11}$　　B. $\frac{34}{11}$　　C. $\frac{33}{11}$　　D. $\frac{34}{10}$

3. 下面有四个数据组,每个组各有三个数据,其中第一个数据为八进制数,第二个数据为十进制数,第三个数据为十六进制数。这四个数据组中三个数据相同的是(　　)。

A. 120 82 50　　B. 144 100 68　　C. 300 200 C8　　D. 1762 1010 3F2

4. 将(2,6,10,17)分别储存到某个地址区间为 0~10 的哈希表中,如果哈希函数 h(x)=(　　),将不会产生冲突,其中 a mod b 表示 a 除以 b 的余数。

A. x mod 11　　B. x^2 mod 11

C. 2x mod 11　　D. $\lfloor\sqrt{x}\rfloor$ mod 11,其中$\lfloor\sqrt{x}\rfloor$表示$\sqrt{x}$向下取整

5. 二分图是指能将定点划分成两个部分,每一部分内的顶点间没有边相连的简单无向图。那么,12 个顶点的二分图至多有(　　)条边。

A. 18　　B. 24　　C. 36　　D. 66

6. 有一个含有 k 个不同的数的数组 $S=<x_1,x_2,\cdots,x_n>$。在 S 中有这样一个数 $x_i(1<i<n)$使得 $x_1<x_2<x_3<\cdots<x_{i-1}<x_i>x_{i+1}>\cdots>x_{n-1}>x_n$,则称这个数 x_i 为数组 S 的“峰顶”,S 就为单峰的。

下面有几行代码,请将 a~e 五处代码补全到算法之中,使得算法正确找到 S 的峰顶。

a. S[mid]<S[mid+1]　　b. S[mid]>S[mid+1]　　c. Search(1,mid-1)

d. Search(mid+1,k)　　e. return S[mid]

```
Search(1,k)
{
  mid=k/2;
  if(S[mid]>S[mid-1]&&_____)
  {
    _____;
  }
  else if(S[mid]>S[mid-1]&&_____)
  {
    _____;
  }
  else _____;
}
```

正确的填空顺序是(　　)。

A. a,c,b,d,e　　B. a,d,b,c,e　　C. b,e,a,d,c　　D. b,e,a,c,d

7. 如果不在快速排序中引入随机化,有可能导致的后果是(　　)。

A. 数组访问越界　　B. 陷入死循环

C. 排序结果错误　　D. 排序时间退化为平方级

8. 二进制数-1101010 的补码是(　　)。

A. 0010101　　B. 10010110　　C. 10010101　　D. 01101010

9. 设二进制数 x 的值是 11001101。若想通过 x&y 运算使 x 中的低度 4 位不变,高 4 位清零,则 y 的二进制数是(　　)。

A. 11110000　　B. 00001100　　C. 00001111　　D. 00000010

10. 循环链表的主要优点是(　　)。

A. 不再需要头指针了

B. 已知某个结点的位置后,能很容易地找到它的直接前驱结点

C. 在进行删除操作后,能保证链表不断开

D. 从表中任一结点出发都能遍历整个链表

11. 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(　　)。

A. O(N * N)　　B. O(N logN)　　C. O(N)　　D. O(1)

12. 若某算法的计算时间表示为递推关系:

$$T(n)=3T(n/4)+n\log_2 n$$

则该算法的复杂度为(　　)。

A. O(n)　　B. $O(n\log_2 n)$　　C. $O(n\log_2^2 n)$　　D. $O(n\log_2^3 n)$

13. 由五个不同的节点构成的无根树有(　　)种。

A. 3125　　B. 125　　C. 32　　D. 1024

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

A. 星期日　　B. 星期五　　C. 星期一　　D. 星期六

15. 一张有 9 个节点的简单无向图最多有(　　)条边。

A. 40　　B. 81　　C. 72　　D. 36

二、阅读程序(程序输入不超过数组或字符串定义的范围;判断题正确的填√,错误的填×;除特殊情况外,判断题每题 1.5 分,选择题每题 3 分,共计 40 分)

1.

```
01  #include<iostream>
02  using namespace std;
03
04  int n,m,i,j,a,b,head,tail,ans;
05  int graph[100][100];
06  int degree[100];
07  int len[100];
08  int queue[100];
09  int main() {
10      cin>>n>>m;
11      for (i=0;i<n;i++)
12          for (j=0;j<n;j++)
13              graph[i][j]=0;
14      for (i=0;i<n;i++)
15          degree[i]=0;
16      for (i=0;i<m;i++) {
17          cin>>a>>b;
18          graph[a][b]=1;
19          (degree[b]++);
20      }
21      tail=0;
```

```
22      for (i=0;i<n;i++)
23          if ((!degree[i])) {
24              queue[tail]=i;
25              tail++;
26          }
27      head=0;
28      while (tail<n) {
29          for (i=0;i<n;i++)
30              if (graph[queue[head]][i]==1) {
31                  (degree[i]--);
32                  if (degree[i]==0) {
33                      queue[tail]=i;
34                      tail++;
35                  }
36              }
37          (++head);
38      }
39      ans=0;
40      for (i=0;i<n;i++) {
41          a=queue[i];
42
43          len[a]=1;
44          for (j=0;j<n;j++)
45              if (graph[j][a]==1 && len[j]+1>len[a])
46                  len[a]=len[j]+1;
47          if ((len[a]>ans))
48              ans=len[a];
49      }
50      cout<<ans<<endl;
51      return 0;
52  }
```

●判断题

(1)若将 19 行“(degree[b]++);”改为“(degree[a]++);”则运行结果不变。(　　)

(2)该程序的时间复杂度是 O(n)。(　　)

(3)代码删除 11 至 15 行,值不变。(　　)

(4)若输入数据为:

4 5
0 2
1 3
0 1
0 3
2 3

则输出 3。(　　)

●选择题

(5)(4 分)第 5 行定义的意义是(　　)。

A. 用邻接矩阵存储图　　B. 记录每个结点的入度

C. 存放拓扑排序结果　　D. 记录以各结点为终点的最长路径长度

(6)这个程序是用了(　　)。

A. 基数排序　　B. 归并排序　　C. 拓扑排序　　D. 堆排序

2.

```
01 #include<iostream>
02 #include<cstring>
03 #define LL long long
04 using namespace std;
05 LL l,r;
06 LL f[12][10][10][2][2][2],a[20];
07 LL Dfs(LL now,LL p,LL pp,LL _4,LL _8,LL top,LL hw) {
08     if (_4 && _8)
09         return 0;
10     if (!now)
11         return hw;
12     if (!top && f[now][p][pp][_4][_8][hw] !=-1)
13         return f[now][p][pp][_4][_8][hw];
14     LL Up=top ? a[now] : 9;
15     LL ret(0);
16     for (LL i=0;i<=Up;++i)
17         ret+=Dfs(now-1,i,p,_4 | (i==4),_8 | (i==8),
18                  top && (i==Up),hw | (i==pp && i==p));
19     if (!top)
20         f[now][p][pp][_4][_8][hw]=ret;
21     return ret;
22 }
23 inline LL Solve(LL x) {
24     LL tot(0);
25     while (x) {
26         a[++tot]=x%10;
27         x /=10;
28     }
29     if (tot !=11)
30         return 0;
31     LL ret(0);
32     for (LL i=1;i<=a[tot];++i)
33         ret+=Dfs(tot-1,i,0,(i==4),(i==8),i==a[tot],0);
34     return ret;
35 }
36 int main() {
37     cin>>l>>r;
38     memset(f,-1,sizeof(f));
39     cout<<Solve(r)-Solve(l-1);
40     return 0;
41 }
```

●判断题

(1)同时包含 4 和 8 的数字都不可能被统计。(　　)

(2)相邻数位中,超过 3 个数位相同的数字都不可能被统计。(　　)

●选择题

(3)(4 分)下列哪个是合法(可能会被统计)的数字?(　　)。

A. 2323234823　　B. 1015400080

C. 23333333333　　D. 10010012022

(4)(5 分)当输入 12121284000 12121285550 时,程序输出结果为(　　)。

A. 5　　B. 457　　C. 455　　D. 6

3.

```
01 #include<iostream>
02 using namespace std;
03 int main() {
04   int n,i,j,x,y,nx,ny;
05   int a[40][40];
06   for (i=0;i<40;i++)
07     for (j=0;j<40;j++) a[i][j]=0;
08   cin>>n;
09   y=0;
10   x=n-1;
11   n=2*n-1;
12   for (i=1;i<=n*n;i++) {
13     a[y][x]=i;
14     ny=(y-1+n)%n;
15     nx=(x+1)%n;
16     if ((y==0 && x==n-1) || a[ny][nx] !=0)
17       y=y+1;
18     else {
19       y=ny;
20       x=nx;
21     }
22   }
23   for (j=0;j<n;j++) cout<<a[0][j]<<" ";
24   cout<<endl;
25   return 0;
26 }
```

●判断题

(1)结果的数字个数为 2n。(　　)

(2)输入 1 时,结果为 1。(　　)

●选择题

(3)若输入 3,则输出是(　　)。

A. 17 24 1 8 15　B. 1 8 24 17 15　　C. 15 24 17 8 1　　D. 24 17 8 1 15

(4)若输入 5,则输出的个数有(　　)。

A. 6　　B. 9　　C. 8　　D. 5

(5)若输入 4 时,则输出结果的和是(　　)。

A. 175　　B. 200　　C. 150　　D. 120

(6)若输入2,则输出(　　)。

A.8 6 1　　B.8 1 6　　C.6 1 8　　D.1 8 6

三、完善程序(单选题,每小题3分,共计30分)

1.(循环比赛日程表)设有N个选手进行循环比赛,其中$N=2^M$,要求每名选手要与其他N-1名选手都赛一次,每名选手每天比赛一次,循环赛共进行N-1天,要求每天没有选手轮空。

输入一个正整数M。输出表格形式的比赛安排表。一行中各数据间用一个空格隔开。

例如输入:3

样例输出:

```
1 2 3 4 5 6 7 8
2 1 4 3 6 5 8 7
3 4 1 2 7 8 5 6
4 3 2 1 8 7 6 5
5 6 7 8 1 2 3 4
6 5 8 7 2 1 4 3
7 8 5 6 3 4 1 2
8 7 6 5 4 3 2 1
```

```
01 #include<cstdio>
02 using namespace std;
03 const int MAXN=1025;
04 int a[MAXN][MAXN];
05 int m;
06 int main() {
07   scanf("%d",&m);
08   int n=1<<m,k=1,half=1;
09   ____①____;
10   while (k<=m) {
11     for (int i=0;i<half;i++) {
12       for (int j=0;j<half;j++) {
13         a[i][____②____]=____③____;
14       }
15     }
16     for (int i=0;i<half;i++) {
17       for (int j=0;j<half;j++) {
18         a[i+half][j]=____④____;
19         a[i+half][j+half]=a[i][j];
20       }
21     }
22     ____⑤____;
23     k++;
24   }
25   for (int i=0;i<n;i++) {
26     for (int j=0;j<n;j++) {
27       printf("%d ",a[i][j]);
28     }
```

```
29      putchar('\n');
30    }
31    return 0;
32  }
```

(1)①处应填(　　)。

A. a[0][1]=1　　B. a[0][0]=1

C. a[1][0]=1　　D. a[1][1]=1

(2)②处应填(　　)。

A. i　　B. i+half　　C. j　　D. j+half

(3)③处应填(　　)。

A. a[i][j]-half　　B. a[i][j+1]-half

C. a[i][j]+half　　D. a[i][j+1]+half

(4)④处应填(　　)。

A. a[i][j+half]　　B. a[i+half][j+half]

C. a[i+half][j]　　D. a[i][j]

(5)⑤处应填(　　)。

A. half/=2　　B. half*=2　　C. half+=2　　D. half--2

2.(并查集)舰队司令莱因哈特率领十万余艘战舰出征,名将杨威利组织麾下三万艘战舰迎敌。在这次决战中,他将巴米利恩星域战场划分成 30000 列,每列依次编号为 1,2,…,30000。之后,他把自己的战舰也依次编号为 1,2,…,30000,让第 i 号战舰处于第 i 列(i=1,2,…,30000)。这是初始阵形。杨威利会多次发布合并指令,将大部分战舰集中在某几列上,实施密集攻击。合并指令为 Mi,j,含义为第 i 号战舰所在的整个战舰队列,作为一个整体(头在前尾在后)接至第 j 号战舰所在的战舰队列的尾部。在杨威利发布指令调动舰队的同时,莱因哈特也会发出一些询问指令:Ci,j。该指令意思是,询问电脑,杨威利的第 i 号战舰与第 j 号战舰当前是否在同一列中,如果在同一列中,那么它们之间布置有多少战舰。你被要求编写程序分析杨威利的指令,以及回答莱因哈特的询问。

```
01  #include<bits/stdc++.h>
02  using namespace std;
03  int fa[30001],front[30001],num[30001],x,y,i,j,n,T,ans;
04  char ins;
05  int find(int n) {
06    if (fa[n]==n)
07      return fa[n];
08    int fn=find(fa[n]);
09    ____①____;
10    return fa[n]=fn;
11  }
12  int main() {
13    cin>>T;
14    for (i=1;i<=30000;++i) {
15      fa[i]=i;
16      ____②____;
17      num[i]=1;
18    }
19    while (T--) {
20      cin>>ins>>x>>y;
```

```
21      int fx=find(x);
22      int fy=find(y);
23      if (ins=='M') {
24        front[fx]+=num[fy];
25        fa[fx]=fy;
26        ____③____;
27        num[fx]=0;
28      }
29      if (ins=='C') {
30        if (____④____)
31          cout<<"-1"<<endl;
32        else
33          cout<<____⑤____<<endl;
34      }
35    }
36    return 0;
37  }
```

(1)①处应填(　　)。

A. front[n]+=front[fa[n]]　　B. front[n]+=front[n]

C. front[fa[n]]+=front[n]　　D. front[fa[n]]+=front[fa[n]]

(2)②处应填(　　)。

A. front[i]=1　　B. front[i]=0

C. front[i]=i　　D. front[fa[i]]=0

(3)③处应填(　　)。

A. num[fx]+=num[fy]　　B. num[fy]+=num[fy]

C. num[fx]+=num[fx]　　D. num[fy]+=num[fx]

(4)④处应填(　　)。

A. fx>fy　　B. fx==fy

C. fx<fy　　D. fx!=fy

(5)⑤处应填(　　)。

A. abs(front[x]-front[y])-1

B. abs(front[x]-front[y])+1

C. abs(front[num[x]]-front[num[y]])-1

D. abs(front[num[x]]-front[num[y]])+1