CSP-J 初赛模拟卷（第 03 套）

入门级 C++ 语言试题

一、单项选择题（共 **15** 题，每题 **2** 分，共计 **30** 分）

1．以下逻辑表达式的值恒为真的是（　）。

A．P ∨ (¬P ∧ Q) ∨ (¬P ∧ ¬Q)
B．Q ∨ (¬P ∧ Q) ∨ (P ∧ ¬Q)
C．P ∨ Q ∨ (P ∧ ¬Q) ∨ (¬P ∧ Q)
D．P ∨ ¬Q ∨ (P ∧ ¬Q) ∨ (¬P ∧ ¬Q)

2．在 8 位二进制补码中，10101011 表示的数是十进制下的（　）。

A．43　　B．-85　　C．-43　　D．-84

3．二进制数 11.01 在十进制下是（　）。

A．3.25　　B．4.125　　C．6.25　　D．11.125

4．把 64 位非零浮点数强制转换成32 位浮点数后，不可能（　）。

A．大于原数　B．小于原数　C．等于原数　D．与原数符号相反

5．提出“存储程序”的计算机工作原理的是（　）。

A．克劳德·香农　　B．戈登·摩尔　　C．查尔斯·巴贝奇　　D．冯·诺依曼

6．如果开始时计算机处于小写输入状态，现在有一只小老鼠反复按照 CapsLock、 字母键 A、字母键 S 和字母键 D 的顺序循环按键，即 CapsLock、A、S、D、CapsLock、A、S、D……，屏幕上输出的第 81 个 字符是字母（　）。

A．A　　B．S　　C．D　　D．a

7．原字符串中任意一段连续的字符所组成的新字符串称为子串。 则字符“AAABBBCCC”共有（　）个不同的非空子串。

A．3　　B．12　　C．36　　D．45

8．广度优先搜索时，需要用到的数据结构是（　）。

A．链表　　B．队列　　C．栈　　D．散列表

9．通常在搜索引擎中，对某个关键词加上双引号表示（　）。

A．排除关键词，不显示任何包含该关键词的结果
B．将关键词分解，在搜索结果中必须包含其中的一部分
C．精确搜索，只显示包含整个关键词的结果
D．站内搜索，只显示关键词所指向网站的内容

10．将（2，6，10，17）分别存储到某个地址区间为0~10 的哈希表中，如果哈希函数=（　），将不会产生冲突，其中表示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)下取整

11．考虑如下递归算法，则调用solve(7)得到的返回结果为（　）。

```
01 solve(n)
02 if n<=1 return 1
03 else if n>=5 return n*solve(n-2)
04 else return n*solve(n-1)
```

A．105　　B．840　　C．210　　D．420

12．给定一个含N个不相同数字的数组，在最坏情况下，找出其中最大或最小的数，至少需要 N-1 次比较操作。则最坏情况下，在该数组中同时找最大与 最小的数至少需要（　）次比较操作。（⌈ ⌉表示向上取整，⌊ ⌋表示向下取整）

A．$\lceil 3N/2 \rceil - 2$　　B．$\lfloor 3N/2 \rfloor - 2$　　C．$2N - 2$　　D．$2N - 4$

13．一家四口人，至少两个人生日属于同一月份的概率是（　）（假定每个人生日属于每个月份的概率相同且不同人之间相互独立）。

A．1/12　　B．1/144　　C．41/96　　D．3/4

14．一棵二叉树的前序遍历序列是ABCDEFG，后序遍历序列是CBFEGDA，则根结点的左子树的结点个数可能是（　）。

A．2　　B．3　　C．4　　D．5

15．2017 年 10 月 1 日是星期日，1999 年 10 月 1 日是（　）。

A．星期三　　B．星期日　　C．星期五　　D．星期二

二、阅读程序（程序输入不超过数组或字符串定义的范围，判断题正确填 **√**，错误填 **×**；共 **18** 小题，共计 **38.5** 分，各小题分值见题目）

(1)

```
01 #include <cstdlib>
02 #include <iostream>
03 using namespace std;
04
05 char encoder[26] = {'C','S','P',0};
06 char decoder[26];
07
08 string st;
09
10 int main()  {
11     int k = 0;
12     for (int i = 0; i < 26; ++i)
13         if (encoder[i] != 0) ++k;
14     for (char x ='A'; x <= 'Z'; ++x) {
15         bool flag = true;
16         for (int i = 0; i < 26; ++i)
17             if (encoder[i] ==x) {
18                 flag = false;
19                 break;
20             }
21         if (flag) {
22             encoder[k]= x;
23             ++k;
24         }
25     }
26     for (int i = 0; i < 26; ++i)
27         decoder[encoder[i]- 'A'] = i + 'A';
28     cin >> st;
29     for (int i = 0; i < st.length( ); ++i)
30         st[i] = decoder[st[i] -'A'];
31     cout << st;
32     return 0;
33 }
```

判断题

16. (1.5分) 输入的字符串应当只由大写字母组成，否则在访问数组时可能越界。（　）

17. (1.5分) 若输入的字符串不是空串，则输入的字符串与输出的字符串一定不一样。（ ）

18. (1.5分) 将第 12 行的 i < 26 改为 i < 16，程序运行结果不会改变。（ ）

19. (1.5分) 将第 26 行的 i < 26 改为 i < 16，程序运行结果不会改变。（ ）

单选题

20. (3分) 若输出的字符串为 ABCABCABCA，则下列说法正确的是（ ）。

A. 输入的字符串中既有 S 又有 P　B. 输入的字符串中既有 S 又有 B
C. 输入的字符串中既有 A 又有 P　D. 输入的字符串中既有 A 又有 B

21. (3分) 若输出的字符串为 CSPCSPCSPCSP，则下列说法正确的是（ ）。

A. 输入的字符串中既有 P 又有 A　B. 输入的字符串中既有 S 又有 R
C. 输入的字符串中既有 A 又有 S　D. 输入的字符串中既无 S 又无 A

(2)

```
01 #include<cstdio>
02 #include<cstring>
03 using namespace std;
04 int n, m;
05 int f[101][101];
06 int F[101][101];
07 int main(){
08     scanf("%d%d",&n,&m);//n的值在1到100之间
09     memset(f,-1,sizeof(f));
10     for(int i = 1; i <=  m; i++) {
11         int u,v,w;//w的值在0到10000之间
12         scanf("%d%d%d",&u,&v,&w);
13         f[u][v]=f[v][u]=w;
14     }
15     for(int k = 1; k <= n; k++)
16         for(int i=1;i <=n; i++)
17             for(int j=1;j<= n; j++)
18                 if(f[i][k]!=-1&&f[k][j]!=-1)
19                     if(f[i][j]==-1||f[i][j]>f[k][j]+f[i][k])
20                         f[i][j]=f[i][k]+f[k][j];
21     int ans=2147483647;
22     for(int i=1;i<=n;i++)
23         for(int j=1;j<=n; j++){
24             for(int x=1;x<=n; x++)
```

```
25                  for(int y=1;y<=n; y++)
26                      F[x][y]=f[x][y];
27              F[i][j]=F[j][i]=0;
28              for(int x=1;x<= n; x++)
29                  for(int y=1;y<=n; y++)
30                      if(F[x][y]==-1||F[x][y]>F[x][i]+F[i][y])
31                          F[x][y]=F[x][i]+F[i][y];
32              for(int x=1;x<=n; x++)
33                  for(int y=1;y<=n; y++)
34                      if(F[x][y]==-1||F[x][y]>F[x][j]+F[j][y])
35                          F[x][y]=F[x][j]+F[j][y];
36              int res=0;
37              for(int x=1;x<=n; x++)
38                  for(int y=1;y<x; y++)
39                      res +=F[x][y];
40              ans = res < ans ? res : ans;
41          }
42      printf("%d\n",ans);
43      return 0;
44  }
```

判断题

22. (1分) 14到16行，将外层到内层的循环变量依次调整为i、j、k，程序的运行的结果不变。（ ）

23. (2分) 这个程序的时间复杂度和m无关。（ ）

24. (2分) 20行的ans如果初始化为10^7时，可能无法得到正确结果。（ ）

25. (2分) 若将第27到30行的部分和31到34行的两个部分互换，程序的运行的结果不变。（ ）

单选题

26. (3分) 若输入数据为4 5/1 2 3/1 3 6/2 3 4/2 4 7/3 4 2(其中“/”为换行符)，则输出为（ ）。

A. 14　　B. 18　　C. 21　　D. 28

27. (3分) 这个程序使用了（ ）算法。

A. Floyd　　B. Dijkstra　　C. Prim　　D. Kruskal

(3)

```
01 #include <iostream>
02 using namespace std;
03 const int V = 101;
04 int n, m, ans, e[V][V];
05 bool visited[V];
06 void dfs(int x, int len)
07 {
08     int i;
09     visited[x] = true;
10     if (len > ans) ans = len;
11     for (i = 1; i <= n; i++)
12         if ((!visited[i]) && (e[x][i] != -1))
13             dfs(i, len + e[x][i]);
14     visited[x] = false;
15 }
16 int main()
17 {
18     int i, j, a, b, c;
19     cin >> n >> m;
20     for (i = 1; i <= n; i++)
21         for (j = 1; j <= n; j++)
22             e[i][j] = -1;
23     for (i = 1; i <= m; i++) {
24         cin >> a >> b >> c;
25         e[a][b] = c;
26         e[b][a] = c;
27     }
28     for (i = 1; i <= n; i++)
29         visited[i] = false;
30     ans = 0;
31     for (i = 1; i <= n; i++)
32         dfs(i, 0);
33     cout << ans << endl;
34     return 0;
35 }
```

输入表示一个没有重边、自环的无向图，顶点编号为 **1** 到 n，且 $1 \leq n \leq 100$、边权为正整数。

判断题

28. (1.5分) 第 19 行的输入中，如果满足 $m = n(n-1)/2$，则第 20～22 行的初始化可以省略。（ ）

29. (1.5分) 将第 31 行改为 `for (i = n; i >= 1; i--)`，程序结果不受影响。（ ）

单选题

30. (1.5分) 若输入如下，则程序输出为（ ）。

```
4 2
1 2 1
3 4 1
```

A. 8　　B. 4　　C. 2　　D. 1

31. (3分) 若输入如下，则程序输出为（ ）。

```
4 6
1 2 1
2 3 1
3 4 1
4 1 1
1 3 1
2 4 1
```

A. 1　　B. 3　　C. 5　　D. 7

32. (3分) 若输入如下，则程序输出为（ ）。

```
4 3
1 2 10
2 3 20
3 1 30
```

A. 30　　B. 50　　C. 90　　D. 120

33. (3分) 若输入如下，则程序输出为（　）。

```
4 6
1 2 10
2 3 20
3 4 30
4 1 40
1 3 50
2 4 60
```

A. 60　　B. 100　　C. 150　　D. 200

三、完善程序（程序输入不超过数组或字符串定义的范围；共 **10** 小题，共计 **29** 分，各小题分值见题目）

(1)

(封禁xxs)现有n个xxs(编号为1到n)，每个xxs都有一个关注者，第i个xxs的关注者是a_i。现在管理员要将其中的一些xxs的账号封禁，但需要注意的是如果封禁了第i个人，那么为了不打草惊蛇，就不能封禁他的关注者a_i。现在想知道最多可以封禁多少个xxs。 输入第一行是一个不超过300000的整数n，第二行是n个1到n的整数表示。 输出一行，一个整数表示答案。

```
01 #include<cstdio>
02 using namespace std;
03 #define MAXN 300005
04 int n, ans=0, a[MAXN], in[MAXN]={0};
05 bool vis[MAXN]={0};
06 void dfs(int cur, int w){
07     if(vis[cur])
08         return;
09     vis[cur]=true;
10     if(w==1) ans++;
11     __(1)__
12     if(__(2)__)
13         dfs(a[cur], __(3)__);
14 }
15 int main(){
16     scanf("%d",&n);
17     for(int i=1;i <=n; i++) {
18         scanf("%d", &a[i]);
19         in[a[i]]++;
20     }
```

```
    for(int i=1;i <= n; i++)
        if(!in[i]) __(4)__;
    for(int i=1;i<=n; i++)
        if(__(5)__) dfs(i,0);
    printf("%d\n",ans);
    return 0;
}
```

34. (3分) (1)处应填

A. a[cur]=cur; B. in[a[cur]]=0; C. in[a[cur]]--; D. in[cur]--;

35. (3分) (2)处应填

A. in[a[cur]] != 0 || w == 1 B. in[a[cur]==0 | w==0
C. in[a[cur]!=0 | w==0 D. in[a[cur]]==0 | w==1

36. (3分) (3)处应填

A. 0 B. 1 C. w D. 1-w

37. (3分) (4)处应填

A. dfs(i,1) B. dfs(i, 0) C. dfs(a[i], 1) D. dfs(a[i], 0)

38. (3分) (5)处应填

A. !in[i] B. in[i] C. !vis[i] D. vis[l]

(2)

完善程序:

（二叉查找树）二叉查找树具有如下性质:每个节点的值都大于其左子树上所有节点的值。小于其右子树上所有节点的值。试判断一棵树是否为二叉查找树。输入的第一行包含一个整数 n，表示这棵树有 n 个顶点，编号分别为 $1,2,\ldots,n$，其中编号为1的为根结点。之后的第1行有三个数value, left_child, right_child，分别表示该节点关键字的值、左子节点的编号、右子节点的编号；如果不存在左子节点或右子节点，则用0代替。输出1表示这棵树是二叉查找树，输出0则表示不是。

```
#include <iostream>
using namespace std;
const int SIZE = 100;
const int INFINITE = 1000000;
struct node
{
```

```
07      int left_child, right_child, value;
08  }; node a[SIZE];
09  int is_bst( int root, int lower_bound, int upper_bound )
10  {
11      int cur;
12      if ( root == 0 )
13          return(1);
14      cur = a[root].value;
15      if ( (cur > lower_bound) && ( __(1)__ ) && (is_bst(
    a[root].left_child, lower_bound, cur ) == 1) && (is_bst( __(2)__,
    __(3)__, __(4)__ ) == 1) )
16          return(1);
17      return(0);
18  }
19  int main()
20  {
21      int i, n; cin >> n;
22      for ( i = 1; i <= n; i++ )
23          cin >> a[i].value >> a[i].left_child >> a[i].right_child;
24      cout << is_bst( __(5)__, -INFINITE, INFINITE ) << endl;
25      return(0);
26  }
```

P/Q	Q为true	Q为false
P为true	P=1&&Q=1	P=1&&Q=0
P为false	P=0&&Q=1	P=0&&Q=0

39. (3分) (1)处应填：_____

40. (3分) (2)处应填：_____

41. (3分) (3)处应填：_____

42. (3分) (4)处应填：_____

43. (2分) (5)处应填：_____