2021 CCF非专业级别软件能力认证第一轮

(CSP-J)入门级C++语言试题 模拟卷 - 10

考生注意事项：

1．全部试题答案均要求写在答卷纸上，写在试卷纸上一律无效。

2．不得使用任何电子设备（如计算器、手机、电子词典等）或查阅任何书籍资料。

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答案）

1．以下关于CSP-J/S的描述错误的是（ ）

A．参加CSP-J/S两组两轮认证均须在网上注册报名，未注册者，无认证成绩。

B．CSP-J/S是中国计算机学会举办的程序设计竞赛。

C．CSP-J/S第二轮实行网上注册、报名，未通过网上报名的认证者可向所在省份特派员申请获得第二轮参加认证的资格。

D．CSP-J/S认证成绩优异者，可参加NOI省级选拔，省级选拔成绩优异者可参加NOI。

2．在8位二进制补码中，10110110表示的是十进制下的（ ）

A．202　　B．74　　C．-202　　D．-74

3．2019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

4．图G是一颗n节点的树，G上有（ ）条边

A．n　　B．2n　　C．n-1　　D．n+1

5．整数在计算机中的二进制表示是（ ）

A．原码　　B．反码　　C．补码　　D．ACSII码

6．有一个长为5的A序列：{3，20，4，6，1}，现通过进行交换其中相邻两个数字的操作进行排序，要将A序列排成从小到大的递增序列最少要进行多少次交换操作？（ ）

A．5　　B．6　　C．7　　D．15

7．7^4和7&4的结果分别为（ ）

A．3 4　　B.4 4　　C．3 3　　D．4 3

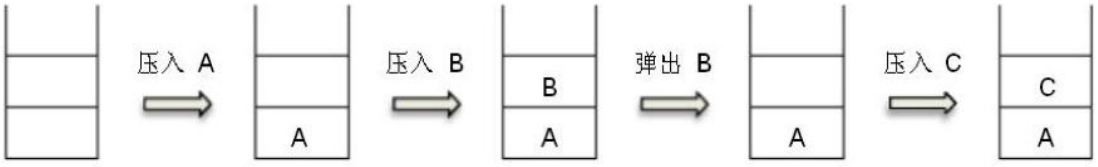
8．一颗6结点二叉树的中序遍历为DBAGECF,后序遍历为DBGEFCAABDCEGF,先序遍历为（ ）

A．DGBEFAC　　B．GBEACFD　　C．ABDCEGF　　D．ABCDEFG

9．一张有9个节点的无向图最多有（ ）条边？

A．40　　B．81　　C．72　　D．36

10. 下图中所使用的数据结构是（ ）。



A.哈希表　　B.栈　　C.队列　　D.二叉树

11．G是一张有n个点m条边的连通图，必须删去（ ）条边才能将其变成一颗n结点的树。

A．1　　B．m-n-1　　C．m+n-1　　D．m-n+1

12．字符串"abcab"本质不同的连续子串(不包含空串)个数为（ ）

A．15　　B．14　　C．13　　D．12

13．十进制小数13.375对应的二进制数是（ ）

A．1101.011　　B．1011.011　　C．1101.101　　D．1010.01

14. 一片容量为 6G 的 SD 卡能储存大约（ ）张大小为 512KB 的数码照片。

A． 10000　　B． 12000　　C． 8000　　D． 16000

15．一家三口人，恰有两人生日在同一天的概率是（ ）。（假设每年都有365天。）

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

二、阅读程序（程序输入不超过数组或字符串定义的范围；判断正确天√，错误填×；除特殊说明外，判断题1.5分，选择题3分，共计40分）

1.

```
#include<iostream>
using namespace std;
int a, b, c;
int main()
{
    cin >> a >> b >> c;
    int t = b;
    b = a, a = t;
    c = a;
    cout << a << " " << b << " " << c;
    return 0;
}
```

约定输入字符串a,b长度均不大于888且不含空格。

- 判断题
  1) 若输入 3 9 1，则输出 9 3 3 。（ ）
  2) 若输入12300400000 3 7，将一定能输出3 12300400000 3。（ ）
  3) 该程序中，头文件#include<iostream> 可以改成#include<cstdio>。（ ）
- 选择题
  4) 若输入3 6 9，输出（ ）。

A. 6 3 6　　B. 9 3 3　　C. 6 9 3　　D. 6 3 3

5) 若将"c=a"改成"c=t",则若输入3 6 9，输出（ ）。

A. 6 3 6　　B. 9 3 3　　C. 6 9 3　　D. 6 3 3

6) 若将"c=a"改成"c=b",则若输入3 6 9，输出（ ）。

A. 6 3 6　　B. 9 3 3　　C. 6 9 3　　D. 6 3 3

2.

```
1  #include <iostream>
2  #include <algorithm>
3  #include <stdio.h>
4  using namespace std;
5  int w[35000], d[35000], dp[35000];
6  int main()
7  {
8      int n, m;
9      scanf("%d%d", &n, &m);
10     for ( int i = 1; i <= n; i++)
11         scanf("%d%d", &w[i], &d[i]);
12     for ( int i = 1; i <= n; i++)
13     {
14         for ( int j = m; j >= w[i]; j--)
15         {
16             dp[j] = max( dp[j], dp[j-w[i]]+d[i] );
17         }
18     }
19     printf("%d\n", dp[m]);
20     return 0;
21 }
```

约定，输入值均为正整数。

- 判断题

1) 上述代码中，双重循环里循环变量 j 的枚举顺序改为 从 w[i] 到 m， 输出结果一定不变。（ ）

2) 上述代码中，双重循环中变量 i 的枚举顺序改为从 n 到 1,输出结果一定不变。（ ）

3) 若输入数据中，1 <= n,m,w[i],d[i] <=30000,则所求答案一定没有溢出。（ ）

4) 若输入数据中，1 <= n,m,w[i] <=30000，1 <= d[i] <=10 0000 0000 ，则所求答案一定没有溢出。（ ）

- 选择题

5) 当输入为：

4 6 1 4 2 6 3 12 2 7

输出为（ ）。

A. 17 B. 28 C. 29 D. 23

6) 上述代码的时间复杂度为（ ）。

A. O(n) B. O(n*n*m) C. O(n*m) D. O(n*m*m)

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;
}
```

约定输入n为区间[1,10000]的正整数。

- 判断题

1) 上述代码实现了对一个长度为n的序列进行排序。（ ）

2) 去掉头文件"#include<cstdio>"后程序仍能正常编译运行。（ ）

3) 去掉"using namespace std;"后程序仍能正常编译运行。（ ）

- 选择题

4) 我们将上述算法称为（ ） 。

A. 插入排序 B. 冒泡排序 C. 选择排序 D. 归并排序

5) 上述代码的时间复杂度为（ ）。

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

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（ ）次（即后面的 tmp=j会执行多少次）。

A. 5 B. 7 C. 3 D. 4

三．完善程序（单选题，每小题3分，共计30份）

1.请完善下面的程序，使用BFS(广度优先搜索)统计一张边权都为1的图中，从给定起点到所有点的距离。

```
#include <algorithm>
#include <cstdio>
#include <vector>
#define N 200020
using namespace std;
int n, m;
vector<int> G[N];
int q[N], hd, tl;
int dis[N];
void BFS( ① )
{
    hd = 1, tl = 0;
    for ( int i = 1; i <= n; i++ ) dis[i] = -1;
    q[++tl] = st, dis[st] = 0;
    while( ② )
    {
        int u = q[ hd++ ];
        for ( int  i = 0; i < G[u].size(); i++)
        {
            int v = G[u][i];
            if( ③ ) continue;
            dis[v] = dis[u]+1;
            q[++tl] = v;
        }
    }
}
int main()
{
    scanf("%d%d", &n, &m); //n结点   m条边
    for (int i = 1; i <= m; i++)
    {
```

```
32          int x, y;
33          scanf("%d%d", &x, &y);  //x与y之间有一条边
34          G[x].push_back(y); //为将元素y插入到G第x行末尾
35          ④;
36      }
37      int st;
38      scanf("%d", &st);
39      BFS( st );
40      for ( ⑤ ) printf("%d", dis[i]);
41  }
```

上述程序中vector<int> G[N]; 用来创建动态二维数组，其第i行的元素个数可根据应用改变。G[u].size()计算第u 行元素个数。

1) 上述程序①处应填（ ）

A.int u; B.int st; C.int &st; D.int st

2) 上述程序②处应填（ ）

A.hd<tl B.hd>tl C.hd<=tl D.hd>=tl

3) 上述程序③处应填（ ）

A. vis[u]==true B.dis[v]<=dis[u]

C.dis[v]!=-1 D.dis[v]>dis[u]

4) 上述程序④处应填（ ）

A. G[y].push_back(x) B. G[y].insert(x)

C. G[y][x]=1 D. G[y].push(y)

5) 上述程序⑤处应填（ ）

A. int j=1;j<=n;j++ B. int i=1;i<=n;i++

C. int i=0;i<n;i++ D.int i=1,i<=n,i++

2.（拓扑排序）给出一张n结点m条边的有向图,求出该图的一个拓扑排序，若无拓扑排序输出-1。

输入：

```
0
000
0
```

第一行两个正整数n,m表示点数与边数。

接下来m行，每行两个正整数x,y;表示结点x->y之间有一条有向边。

输出:一个拓扑序，按拓扑序输出点的编号。若拓扑序不唯一，输出任意一个均可。若无拓扑序，输出-1。

关于拓扑序的例子,如学校里有A,B,C,D四门课程,要求课程B,C必须在学习课程A之后才能学习，课程D必须在学习课程B,C之后学习。则序列A B C D与序列A C B D均是合理的拓扑序，而序列A B D C与序列 B A C D等均不是拓扑序。

即在安排某课程时，其前置课程必须全部学习完毕。

试补全程序。

```
1  #include <algorithm>
2  #include <cstdio>
3  #include <vector>
4  #define N 200020
```

```
using namespace std;
int n, m;
vector<int> G[N];
int q[N], hd, tl;
int du[N];
int ans[N], tot;
void topo()
{
    hd = 1, tl = 0;
    for ( int i = 1; i <= n; i++ ) if( ① ) q[++tl] = i;
    while( hd <= tl )
    {
        int u = q[hd++];
        ans[++tot] = u;
        for ( int i = 0; ② ; i++ )
        {
            int v = G[u][i];
            du[v]--;
            if(!du[v])  ③ ;
        }
    }
    if(tot != n) puts("-1");
    else
    {
        for ( int i = 1; i <= n; i++ )
            printf("%d ", ans[i]);
    }
}
int main() {
    scanf("%d%d", &n, &m);
    for ( int i=1; i <= m; i++ )
    {
        int x, y;
        scanf("%d%d", &x, &y);
         ④ ;
         ⑤ ;
    }
    topo();
    return 0;
}
```

1) 上述程序①处应填（ ）

A. du[i]　　B. q[i]　　C. hd<=tl　　D. !du[i]

2) 上述程序②处应填（ ）

A. i<=n　　B. i<n　　C. i<G[u].size()　D. i<=G[u].size()

3) 上述程序③处应填（ ）

A. q[++tl]=v　　B. q[tl++]=v

C. q[++hd]=v　　D. q[hd++]=v

4) 上述程序④处应填（ ）

A. G[y].push_back(x)　B. G[x].push_back(y)

C. G[x].push(y)　　D. G[y].push(x)

5) 上述程序⑤处应填（ ）

A. G[y].push_back(x)　B. G[y].push(x)

C. du[y]++　　D. du[x]++