全国入门组 CSP-J 初赛模拟试题（4）

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．不同类型的存储器组成了多层次结构的存储器体系，按存取速度从快到慢排列的是（ ）

A.快存/辅存/主存　　B.外存/主存/辅存　　C.快存/主存/辅存　　D.主存/辅存/外存

2．RAM 中的信息是(　　)。

A.生产厂家预先写入的　　B.计算机工作时随机写入的

C.防止计算机病毒侵入所使用的　　D.专门用于计算机开机时自检用的

3．在 24*24 点阵的字库中，汉字“一”与“编”的字模占用字节数分别是（）。

A.72、72　　B.32、32　　C.32、72　　D.72、32

4．计算机的运算速度取决于给定的时间内，它的处理器所能处理的数据量。处理器一次能处理的数据量叫字长。已知 64 位的奔腾处理器一次能处理 64 个信息，相当于（）字节

A.8 个　　B.1 个　　C.16 个　　D.2 个

5．在计算机领域中，通常用英文单词“BYTE”来表示（）。

A 字　　B.字长　　C.二进制位　　D.字节

6．GB2312-80 规定了一级汉字 3755 个，二级汉字 3008 个，其中二级汉字字库中的汉字是以（）为序排列的。（翻书记一记）

A.以笔划的多少　　B.以部首　　C.以 ASCII 码　　D.以机内码

7．设栈 S 的初始状态为空，现有 5 个元素组成的序列{1，2，3，4, 5}，对该序列在 S 栈上依次进行如下操作(从序列中的 1 开始，出栈后不再进栈):进栈、进栈、进栈、出栈进栈、出栈、进栈。试问出栈的元素序列是（）。

A.{5,4,3.2,1}　　B.{2.1}　　C.{2,3}　　D.{3,4}

8．设循环队列中数组的下标范围是 n,其中头尾指针分别是 f 和 r，则其元素个数是（）。

A.r-f　　B.r-f+1　　C.(r-f) % n+1　　D.(r-f+n) % n

9．电线上停着两种鸟(A, B)，可以看出两只相邻的鸟就将电线分为了一个线段。这些线段可公为两类: 一类是两端的小鸟相同；另一类是两端的小鸟不相同。已知:电线上两个顶点上正好停着相同的小鸟，试问两端为不同小鸟的线段数目一定是（）。

A.奇数　　B.偶数　　C.可奇可偶　　D.数目固定

10．从未排序序列中挑选元素，并将其依次放入已排序序列(初始时为空)的一端，这种排序方法称为（）。

A.插入排序　　B.归并排序　　C.选择排序　　D.快速排序

11．对一个满二叉树，m 个树叶，1 分枝结点，n 个结点，则（）。

A.n=l+m　　B.l+m=2n　　C.m=l-1　　D.n=2l-1

12．下列哪个软件不是操作系统软件的名字（）。

A.WindowsXP　　B.Arch/Info　　C.Linux　　D.OS/2

13．下列哪个不是个人计算机的硬件组成部分（）。

A.主板　　B.虚拟内存　　C.总线　　D.硬盘

14．已知元素(8，25，14，87，51, 90，6，19， 20)，问这些元素以怎样的顺序进入栈，才能使出栈的顺序满足:8 在 51 前面:90 在 87 的后面;20 在 14 的后面:25 在 6 的前面；19 在 90 的后面。（）。

A.20,6,8,51,90,25,14,19.87　　B.51,6,19,20,14,8,87,90,25

C.19,20,90,7,6,25,51.14,87　　D.6,25,51,8,20,19,90,87,14

15．假设我们用 d=(a1,a2....,a5), 表示无向图 G 的 5 个顶点的度数，下面给出的哪组 d 值合理（）。

A.{2,2,2,2,2}　　B.{1,2,2,1.1}　　C.{3,3,3,2,2}　　D.{5,4,3,2,1}

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

1、

```
1    #include<iostream>
2    using namespace std;
3    #include<cmath>
4    bool IsPrime(int num)
5    {
6        for (int i = 2; i <= sqrt(num); i++)
7        {
8            if (num%i == 0)
9            {
10               return false;
11           }
12       }
13       return true;
14   }
15   int main()
16   {
17       int num = 0;
18       cin >> num;
19       if (IsPrime(num))
20       {
21           cout << "YES" << endl;
22       }
23       else
24       {
25           cout<< "NO"<<endl;;
26       }
27       return 0;
28   }
```

判断题

1) (1 分)第 18 行输入 97 时，输出为“NO” (不含引号)。(　　)

2) (1 分)第 18 行输入 119 时，输出为“YES” (不含引号)。(　　)

3) 若将第 6 行的“<=”改成“<”，程序输出的结果一定不会改变。(　　)

4) 当程序执行第 13 行时，i 值为 sqrt(num) 。(　　)

选择题

5)（3 分）最坏情况下，此程序的时间复杂度是（　　）。

A.O(num)　　B.O(num^2)　　C.O(sqrt(num))　　D.O(lognum)

6) 若输入的 num 为 20 以内的正整数，则输出为 “YES” 的概率是（　　）。

A.0.45　　B.0.4　　C.0.5　　D.0.35

2、

```
1    #include<bits/stdc++.h>
2    using namespace std;
3    const int mod=2048;
4    long long c,n;
5    long long kasumi(long long x,long long mi) {
6        long long res=1;
7        while(mi) {
8            if(mi&1) {
9                res=(res*x)%mod;
10           }
11           x=(x*x)%mod;
12           mi>>=1;
13       }
14       return res;
15   }
16   int main() {
17       cin>>n>>c;
18       if(n==3) {
19           printf("%lld",c*(c-1));
20           return 0;
21       }
22       long long ans=((kasumi(c-1,n)+(c-1)*kasumi(-1,n))%mod+mod)%mod;
23       cout<<ans ;
24       return 0;
25   }
```

判断题

1) (1 分)将第 9 行和第 11 行的括号去掉，程序输出结果一定不变。(　　)

2) (1 分)将第 12 行的“mi>>=1”改为“mi*=0.5”，程序输出结果一定不变。(　　)

3) 若输入为“4 4”，则输出为“78”。(　　)

4) 此程序的时间复杂度为 O(logn)。(　　)

选择题

5）若输入为“3 4”，则输出为（　　）。

A.8　　B.12　　C.18　　D.19

6）kasumi (2046,13) 的返回值为（）。

A.0　　B.2022　　C.2　　D.2024

3、

```
1    #include<cstdio>
2    int n,r,num[ 10000];
3    bool mark[10000];
4    void print()
5    {
6        for(int i=1; i<=r; i++)
7            printf("%d ",num[i]);
```

```
8           printf("\n");
9     }
10    void search(int x)
11    {
12          for(int i=1; i<=n; i++)
13                if(!mark[i])
14                {
15                      num[x]=i;
16                      mark[i]=true;
17                      if(x==r) print();
18                      search(x+1);
19                      mark[i]=false;
20                }
21    }
22    int main()
23    {
24          scanf("%d%d",&n,&r);
25          search(1);
26    }
```

判断题

1) (1 分)程序结束时，对任意 1≤i≤n， mark[i]=0。(　　)

2) (2 分)若 n<r，则程序无输出。 (　　)

3) (2 分)若输入为“4 3”，则输出中数字 1 和 2 的个数不同。(　　)

4) (2 分)此程序的时间复杂度为 O(n)。(　　)

选择题

5) 若输入为“6 3”，则函数 print 的执行次数为（　　）。

A.60　　B.120　　C.6　　D.720

6) 若输入为“7 4”，则输出的最后一行为（　　）。

A.4 5 6 7　　B.7 6 5 4　　C.4 3 2 1　　D.1 2 3 4

三、完善程序（单选题，每题 3 分，共计 30 分）

1、克鲁斯卡尔求最小生成树思想:首先将 n 个点看做 n 个独立的集合，将所有边快排(从小到大)。然后，按排好的顺序枚举每一条边， 判断这条边连接的两个点是否属于一个集合。若是，则将这条边加入最小生成树，并将两个点所在的集合合并为一个集合。若否，则跳过。直到找到 n-1 条边为止。

```
#include<iostream>
#include<algorithm>
using namespace std;
struct point {
      int x;
      int y;
      int v;
};
point a[10000];
```

```
int cmp(const point &a,const point &b) {
    if(___(1)___) return 1;
    else return 0;
}
int fat[101];
int father(int x) {
    if(fat[x] !=x) return fat[x]= ___(2)___;
    else return fat[x];
}
void unionn(int x,int y) {
    int fa=father(x);
    int fb=father(y);
    if(fa!=fb) fat[fa]=fb;
}
int main() {
    int i,j,n,m, k=0, ans=0,cnt=e;
    cin>>n;
    for(i=1; i<=n; i++)
    for(j=1; j<=n; j++)
    {
        cin>>m;
        if(m!=0) {
            k++;
            a[k].x=i;
            a[k].y=j;
            a[k].v=m;
        }
    }
    sort(a+1,a+1+k; ___(3)___ );
    for(i=1; i<=n; i++) {
        fat[i]=i;
    }
    for(i=1; i<=k; i++) {
        if(father(a[i].x)!=___(4)___) {
            ans+=a[i].v;
            unionn(a[i].x,a[i].y);
            cnt++;
        }
        if(___(5)___) break ;
    }
    cout<<ans;
    return 0;
}
```

选择题

1）①处应填（）

A.a.v < b.v　　B.a.v > b.v　　C..a.v >= b.v　　D.a.v <=b.v

2）②处应填（）

A.father(x)　　B. father (fat[x])　　C.fat(father[x])　　D.x

3）③处应填（）

A.algorithm　　B.point　　C.cmp　　D.sizeof(a)

4）④处应填（）

A.a[i].y　　B.father(a[i].y)　　C.fat[a[i].y]　　D.a[i].x

5）⑤处应填（）

A.cnt > 0　　B.i == 1　　C.ans == n-1　　D.cnt == n-1

2、欧拉回路问题由七桥问题而来，其基本问题是是否能一次性不重复地走遍这七座桥，转换为数学问题中的图论就是指的是从图中的一个顶点出发，是否能够一次性不回头地走遍所有的边，算法代码如下：

```
#include <iostream>
#include <ctime>
using namespace std;
int G[5][5];
int visited[5][5];
int n = 5;
void euler(int u) {
	for (int v = e; v < n; v++) {
		if (G[u][v] & ___(1)___) {
			cout << u <<"->" << v << endl;
			visited[u][v]= visited[v][u] = ___(2)___;
			 ___(3)___
		}
	}
}
int main() {
	G[1][2] = G[2][1] = G[1][3] =    ___(4)___= 1;
	G[2][4] = G[4][2] = G[3][4] =    ___(5)___= 1;
	euler(1);
	return 0;
}
```

选择题

1) ①处应填( )

A.G[v][u]　　B.!visited[u][v]　　C.visited[u][v]　　D.visited[v][u]

2) ②处应填( )

A. 1　　B.0　　C.u　　D.v

3) ③处应填( )

A.euler(v);　　B.euler(u);　　C.G[u][v] = 0;　　D.G[v][u] = 0;

4) ④处应填( )

A.G[0][1]　　B.G[1][0]　　C.G[3][1]　　D.G[0][3]

5) ⑤处应填( )

A.G[0][2]　　B.G[2][0]　　C.G[2][1]　　D.G[4][3]