

入门级 CSP-J 第 13 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. 下列关于解释程序和编译程序的四条叙述,其中正确的是()。
 - A. 解释程序产生目标程序
 - B. 编译程序产生目标程序
 - C. 解释程序和编译程序都产生目标程序
 - D. 解释程序和编译程序都不产生目标程序
2. 十进制数(-123)的原码表示为()。
 - A. 11111011
 - B. 10000100
 - C. 1000010
 - D. 01111011
3. 网络管理员排查无法访问 www.cisco.com 的故障,发现在浏览器中键入 web 服务器的 IP 地址可以访问网页,那故障应该归咎于哪个应用层协议()。
 - A. DHCP
 - B. DNS
 - C. HTTP
 - D. POP3
4. 以下程序段执行完毕后,输出的结果是()。

```
#include<bits/stdc++.h>
```

```
using namespace std;
```

```
int prime(int n){
```

```
    int i,y=1;
```

```
    for(i=2;i*i<=n;i++){
```

```
        if(n%i==0){
```

```
            y=0;
```

```
            break;
```

```
        }
```

```
    }
```

```
    return y;
```

```
}
```

```
int main(){
```

```
    int m=90,n=100;
```

```
    while(m!=(n+1)){
```

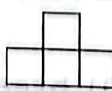
```
        if(prime(m)==1)cout<<m<<" ";
```

```
        m++;
```

```
    }
```

```
    return 0;
```

```
}
```

- A. 91
 - B. 97
 - C. 91 97
 - D. 91 95 97
5. 若已知一个栈的入栈序列是 1,2,3...n,其输出序列为 p1,p2,p3...pn,若 p1=n,则 pi 为()。
 - A. i
 - B. n-i
 - C. n-i+1
 - D. 不确定
 6. 使用分治法求解不需要满足的条件是()。
 - A. 子问题必须是一样的
 - B. 子问题不能够重复
 - C. 子问题的解可以合并
 - D. 原问题和子问题使用相同的方法解
 7. 如图,在 5×7 的方格表中有多少个形状为“”的图形?

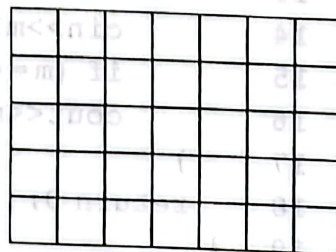
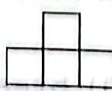
()。

A. 42

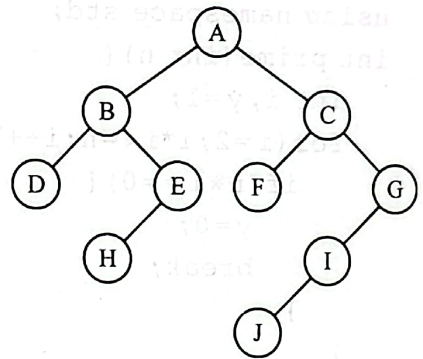
B. 15

C. 76

D. 20



8. 堆的形状是一棵()。
 A. 二叉排序树 B. 满二叉树 C. 完全二叉树 D. 平衡二叉树
9. 围着一张圆桌给 3 名男生, 6 名女生安排座位, 座位没有编号。如果两名男生之间恰有两名女生, 共有多少种安排座位的方法()。
 A. 392880 B. 1440 C. 2160 D. 720
10. 100 以内的质数有()个。
 A. 25 B. 26 C. 27 D. 28
11. 4 名嘉宾和 2 名领导站成一排参加剪彩, 其中领导不能相邻, 则站位方法总数为()。
 A. 720 B. 480 C. 120 D. 60
12. 一盒围棋子, 4 个 4 个数多 3 个, 6 个 6 个数多 5 个, 15 个 15 个数多 14 个, 棋子在 150~200 个, 棋子共几个? ()。
 A. 167 B. 179 C. 194 D. 以上都不对
13. 数据结构中, “先进先出”是()结构的特征。
 A. 队列 B. 栈 C. 线性表 D. 树
14. 右图所示二叉树的中序序列是()。
 A. DHEBAFIJCG
 B. DHEBAFJICG
 C. DBHEAFCJIG
 D. DBHEAFJICG
15. 表达式的后缀表达式 $9\ 3\ 1-3\ *+10\ 2\ /+$ 的值为()。
 A. 45
 B. 19
 C. 20
 D. 51



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×; 除特殊说明外, 判断题每题 1.5 分, 选择题每题 3 分, 共计 40 分)

1.

```

01 #include<bits/stdc++.h>
02 using namespace std;
03 long long normalPower(long long base, long long power) {
04     long long result=1;
05     for (int i=1; i<=power; i++) {
06         result=result*base;
07         result=result%1000;
08     }
09     return result%1000;
10 }
11 int main() {
12     long long m,n;
13     while (true) {
14         cin>>m>>n;
15         if (m==0 && n==0) break;
16         cout<<normalPower(m,n)<<endl;
17     }
18     return 0;
19 }
    
```


● 判断题

- (1) 输出的结果为三位数。()
- (2) 将 05 行的 \leq 改为 $<$ 则输出结果不变。()
- (3) 将第 13 行 true 改为 1, 程序运行结果不会改变。()
- (4) 将第 15 行删除, 程序运行结果不会改变。()

● 选择题

- (5) 如果输入 2 和 12, 则输出结果为多少()。
- A. 4 B. 96 C. 096 D. 4096
- (6) (4 分) 这个算法的时间复杂度为()。
- A. $O(m * n)$ B. $O(n)$ C. $O(m)$ D. $O(m^n)$

2.

```

01 #include<bits/stdc++.h>
02 using namespace std;
03 const int N=1100;
04 int n,k,m;
05 int a[N];
06 int vis[N];
07 int main()
08 {
09     cin>>n>>k>>m;
10     int cnt=0;
11     int num=0;
12     while(cnt<n-1)
13     {
14         int cnt2=1;
15         while(cnt2<=n)//遍历数组
16         {
17             int pos=k+cnt2;
18             if(pos>n)
19             {
20                 pos=pos-n;
21             }
22             if(!vis[pos])
23             {
24                 num++;
25                 if(num%m==0)
26                 {
27                     cnt++;
28                     vis[pos]=1;
29                     if(cnt==n-1)
30                         break;
31                 }
32             }
33             cnt2++;
34         }
35     }

```

```

36     for(int i=1;i<=n;i++)
37     {
38         if(!vis[i])
39         {
40             cout<<i<<endl;
41             break;
42         }
43     }
44     return 0;
45 }

```

●判断题

- (1) 上述代码中,将第 29 行 `==` 修改为 `>=`,输出结果一定不变。()
- (2) 上述代码中,将第 29、30 行删除,输出结果也一定相同。()
- (3) 上述代码中,输入的 `k` 值可以大于 `n`。()
- (4) 上述代码中,输入的 `m` 值可以大于 `n`。()

●选择题

- (5) 当输入为:12 3 8,输出为()。
A. 1 B. 3 C. 7 D. 9
- (6) (4 分) 上述代码中,利用数组模拟的是()。
A. 队列 B. 环 C. 树 D. 以上都不是

3.

```

01 #include<bits/stdc++.h>
02 using namespace std;
03 int select_arr(int arr[],int len,int arr_value)
04 {
05     while (1)
06     {
07         int left=0;           //数组的左侧下标
08         int right=len-1;      //数组的右侧下标
09         while (left<=right)
10         {
11             int mid=(left+right)/2; //定义中间位的下标
12             int mid_value=arr[mid]; //定义中间值的基准值
13             if (mid_value==arr_value) //如果基准值正好等于要查找的值
14             {
15                 return mid;
16             }
17             else if (mid_value>arr_value)
18             {
19                 right=mid-1;
20             }
21             else if (mid_value<arr_value)
22             {
23                 left=mid+1;
24             }
25         }

```



```

26     return -1;
27 }
28 }
29 int main()
30 {
31     int arr[10]={ 1,3,5,7,9,10,16,46,88,91 };
32     int weizhi=select_arr(arr,10,16);
33     cout<<weizhi;
34     return 0;
35 }

```

●判断题

- (1) 数组 arr[] 的值可以为负数。()
- (2) 若第 31 行代码为 int arr[10]={ 1, 3, 7, 5, 9, 10, 16, 46, 88, 91 }, 输出结果是一样的。()
- (3) 数组数值越大, 排序效率越低。()
- (4) 当数组数据量越大时, 和顺序查找相比优势越明显。()

●选择题

- (5) (4 分) 程序输出结果为()。

A. 4

B. 5

C. 6

D. 7

- (6) (4 分) 该程序的算法为()。

A. 顺序查找

B. 二分查找

C. 哈希查找

D. 分块查找

三、完善程序 (单选题, 每题 3 分, 共计 30 分)

1. 有形如: $ax^3+bx^2+cx+d=0$ 这样的一个一元三次方程。给出该方程中各项的系数(a, b, c, d 均为实数), 并约定该方程存在三个不同实根(根的范围在 -100 至 100 之间), 且根与根之差的绝对值 ≥ 1 。要求由小到大依次在同一行输出这三个实根(根与根之间留有空格), 并精确到小数点后 2 位。

提示: 记方程 $f(x)=0$, 若存在 2 个数 x_1 和 x_2 , 且 $x_1 < x_2$, $f(x_1) * f(x_2) < 0$, 则在 (x_1, x_2) 之间一定有一个根。

试补全程序。

```

01 #include<bits/stdc++.h>
02 using namespace std;
03 double a,b,c,d;
04 double f( ① ) //返回一元三次方程值
05 {
06     return 1.0*a*pow(x,3)+b*pow(x,2)+c*pow(x,1)+d;
07 }
08
09 double mid(double x,double y)
10 {
11     double st=x,en=y,mid=(st+en)*1.0/2;
12     while((double)fabs(f(mid))>0.000001)
13     {
14         if(f(st)*f(mid)>0)
15         {
16             st= ② ;
17         }else if(f(en)*f(mid)>0){

```

```

18         en=mid;
19     }
20     mid=( ③ ) *1.0/2;
21 }
22 return mid; //找到根
23 }
24
25 int main()
26 {
27     cin>>a>>b>>c>>d;
28     double x=-100,y=-100;
29     int num=0;
30     double c[3];
31     while( ④ )
32     {
33         if(f(x)*f(y)>0)
34         {
35             y+=0.9;
36         }else{
37             c[num]= ⑤ ;
38             num++;
39             x=y; //继续往后遍历
40         }
41     }
42     printf("%.21f%.21f%.21f",c[0],c[1],c[2]);
43     return 0;
44 }

```

(1)①处应填()。

A. int x B. float x C. double x D. long long x

(2)②处应填()。

A. x B. y C. mid D. en

(3)③处应填()。

A. x+y B. y C. st+en D. en

(4)④处应填()。

A. num<1 B. num<2 C. num<3 D. num<4

(5)⑤处应填()。

A. f(x) * f(y) B. x C. mid(x,y) D. y

2. (dijkstra)给定一个有 n 个顶点(从 1 到 n 编号), m 条边的有向图(其中某些边权可能为负,但保证没有负环)。请你计算从 1 号点到其他点的最短路。

```

01 #include<bits/stdc++.h>
02 using namespace std;
03 int n,m;
04 const int ff=0x3f3f3f3f;
05 long long a[20020];
06 struct node
07 {

```



```

08     int u,v,w;
09 }mp[200002];
10 void dj()
11 {
12     a[1]=0;
13     for(int i=2; ① ;i++)
14     {
15         int k=-1;
16         for(int j=1; ② ;j++)
17         {
18             if(a[mp[j].u]+mp[j].w<a[mp[j].v])
19             {
20                 a[mp[j].v]= ③ ;
21                 k=1;
22             }
23         }
24         if( ④ )
25             break;
26     }
27     for(int i=2;i<=n;i++)
28         cout<< ⑤ <<endl;
29 }
30 int main()
31 {
32     cin>>n>>m;
33     memset(a,ff,sizeof(a));
34     for(int i=1;i<=m;i++)
35     {
36         cin>>mp[i].u>>mp[i].v>>mp[i].w;
37     }
38     dj();
39     return 0;
40 }

```

(1) ①处应填()。

- A. $i < n$ B. $i <= n$ C. $i < m$ D. $i <= m$

(2) ②处应填()。

- A. $j < n$ B. $j <= n$ C. $j < m$ D. $j <= m$

(3) ③处应填()。

- A. $a[mp[j].u]+mp[j].w$ B. $a[mp[j].v]+mp[j].w$
 C. $a[mp[i].u]+mp[i].w$ D. $a[mp[i].v]+mp[i].w$

(4) ④处应填()。

- A. $k = 0$ B. $k = 1$ C. $k = -1$ D. $k = 2$

(5) ⑤处应填()。

- A. $mp[i].v$ B. $a[i]$ C. $mp[i].u$ D. $mp[a[i]].u$