CSP-J 初赛模拟卷（第 10 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．中国计算机学会于（　）年创办全国青少年计算机程序设计竞赛。

A．1983　　B．1984　　C．1985　　D．1986

2．目前主流的计算机储存数据最终都是转换成（　）数据进行储存。

A．二进制　　B．十进制　　C．八进制　　D．十六进制

3．以下奖项与计算机领域最相关的是（　）。

A．奥斯卡奖　　B．图灵奖　　C．诺贝尔奖　　D．普利策奖

4．下列四个不同进制的数中，与其它三项数值上不相等的是

A．(269)16　B．(617)10　C．(1151)8　D．(1001101011)2

5．链表不具有的特点是（　）。

A．可随机访问任一元素　　B．不必事先估计存储空间

C．插入删除不需要移动元素　D．所需空间与线性表长度成正比

6．下面的故事与（　）算法有着异曲同工之妙。 从前有座山，山里有座庙，庙里有个老和尚在给小和尚讲故事：“从前有座 山，山里有座庙，庙里有个老和尚在给小和尚讲故事：‘从前有座山，山里有座庙，庙里有个老和尚给小和尚讲故事……’”

A．枚举　　B．递归　　C．贪心　　D．分治

7．对于有n个顶点、m条边的无向联通图（m > n），需要删掉（　）条边才能使其成为一棵树。

A．n - 1　　B．m - n　　C．m - n - 1　　D．m - n + 1

8．5 个小朋友并排站成一列，其中有两个小朋友是双胞胎，如果要求这两个双胞胎必须相邻，则有（　）种不同排列方法?

A．48　　B．36　　C．24　　D．72

9．如果一棵二叉树只有根结点，那么这棵二叉树高度为1。请问高度为5的完全二叉树有（ ）种不同形态?

A．16　　B．15　　C．17　　D．32

10．有五副不同颜色的手套（共 10 只手套，每副手套左右手各 1 只），一次性从中取 6 只手套，请问恰好能配成两副手套的不同取法有（ ）种。

A．120　　B．180　　C．150　　D．30

11．一副纸牌除掉大小王有52张牌，四种花色，每种花色13张。假设从这52张牌中随机抽取13张纸牌，则至少（ ）张牌的花色一致。

A．4　　B．2　　C．3　　D．5

12．设 A 是一个实数的数组，考虑下面的递归算法：

```
XYZ (A[1..n])
if n= 1 then return A[1]
else temp ← XYZ (A[1..n-1])
    if temp < A[n]
    then return temp
    else return A[n]
```

请问算法XYZ的输出是什么？（ ）。

A．A数组的平均　　B．A数组的最小值　　C．A数组的中值　　D．A数组的最大值

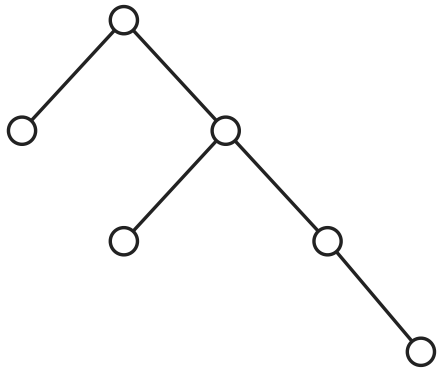
13．假设一棵二叉树的后序遍历序列为DGJHEBIFCA,中序遍历序列为DBGEHJACIF,则其前序遍历序列为（ ）。

A．ABCDEFGHIJ　　B．ABDEGHJCFI　　C．ABDEGJHCFI　　D．ABDEGHJFIC

14．一些数字可以颠倒过来看，例如0、1、8颠倒过来还是本身，6颠倒过来是9，9颠倒过来看还是6,其他数字颠倒过来都不构成数字。 类似的，一些多位数也可以颠倒过来看，比如106颠倒过来是901。假设某个城市的车牌只由5位数字组成，每一位都可以取0到9。 请问这个城市最多有多少个车牌倒过来恰好还是原来的车牌？（ ）

A．60　　B．125　　C．75　　D．100

15. 一棵二叉树如右图所示，若采用顺序存储结构，即用一维数组元素存储该二叉树中的结点（根结点的下标为1,若某结点的下标为i，则其左孩子位于下标2i处、右孩子位于下标 $2i+1$ 处），则该数组的最大下标至少为（　）。



A. 6　　B. 10　　C. 15　　D. 12

二、阅读程序（程序输入不超过数组或字符串定义的范围，判断题正确填 √，错误填 ×；共 18 小题，共计 36 分，各小题分值见题目）

(1)

```
#include <cstdlib>
#include <iostream>
using namespace std;

char encoder[26] = {'C','S','P',0};
char decoder[26];

string st;

int main()  {
    int k = 0;
    for (int i = 0; i < 26; ++i)
        if (encoder[i] != 0) ++k;
    for (char x ='A'; x <= 'Z'; ++x) {
        bool flag = true;
        for (int i = 0; i < 26; ++i)
            if (encoder[i] ==x) {
                flag = false;
                break;
            }
        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 <iostream>
02 using namespace std;
03
04 int n;
05 int d[1000];
06
07 int main() {
08     cin >> n;
09     for (int i = 0; i < n; ++i)
10         cin >> d[i];
11     int ans = -1;
12     for (int i = 0; i < n; ++i)
13         for (int j = 0; j < n; ++j)
14             if (d[i] < d[j])
15                 ans = max(ans, d[i] + d[j] - (d[i] & d[j]));
16     cout << ans;
17     return 0;
18 }
```

假设输入的n和d[i]都是不超过10000的正整数，完成下面的判断题和单选题：

判断题

22. (1.5分) n必须小于1000，否则程序可能会发生运行错误。（ ）

23. (1.5分) 输出一定大于等于0。（ ）

24. (1.5分) 若将第13行的“j = 0”改为“j = i + 1”，程序输出可能会改变。（ ）

25. (1.5分) 将第14行的“d[i] < d[j]”改为“d[i] != d[j]”，程序输出不会改变。（ ）

单选题

26. (3分) 若输入n为100，且输出为127，则输入的d[i]中不可能有（ ）。

A. 127　　B. 126　　C. 128　　D. 125

27. (3分) 若输出的数大于0，则下面说法正确的是（ ）。

A. 若输出为偶数，则输入的d[i]中最多有两个偶数
B. 若输出为奇数，则输入的d[i]中至少有两个奇数
C. 若输出为偶数，则输入的d[i]中至少有两个偶数
D. 若输出为奇数，则输入的d[i]中最多有两个奇数

(3)

```
01 #include<iostream>
02 using namespace std;
03 int n,m,i,j,p,k;
04 int a[100],b[100];
05 int main()
06 {
07     cin>>n>>m;
08     a[0]=n;i=0;p=0;k=1;
09     do{
10         for(j=0;j<i;j++)
11             if(a[i]==a[j])
12             {
13                 p=1;k=j;break;
14             }
15         if(p) break;
16         b[i]=a[i]/m;
17         a[i+1]=a[i]%m*10;
18         i++;
19     }while(a[i]!=0);
20     cout<<b[0]<<".";
21     for(j=1;j<k;j++) cout<<b[j];
22     if(p) cout<<"(";
23     for(j=k;j<i;j++) cout<<b[j];
24     if(p) cout<<") ";
25     cout<<endl;
26     return 0;
27 }
```

判断题

28. (1.5分) 程序输入的n和m不能相等。（ ）

29. (1.5分) 程序输入的m不能等于0。（ ）

30. (1.5分) 第9~19行的do…while循环一共有2个出口。（ ）

31. (1.5分) 数组a和b中的数值都小于或等于n。（ ）

单选题

32. (3分) 若输入数据为“11 8”,则输出结果为（ ）

A. 0.375　　B. 0.(375)　　C. 1.375　　D. 1.(375)

33. (3分) 若输人数据为“5 13”，则输出结果为（ ）。

A. 0.386514　　B. 0.(386514)　　C. 0.384615　　D. 0.(384615)

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)

完善程序:

（郊游活动）有 n 名同学参加学校组织的郊游活动，已知学校给这 n 名同学的郊游总经费为 A 元，与此同时第 i 位同学自己携带了 M_i 元。为了方便郊游，活动地点提供 $B(\geq n)$ 钢自行车供租用，租用第 i 钢自行车的价格为 C_i 元，每个同学可以使用自己携带的钱或者学校的郊游经费。为了方便账务管理，每位同学只能为自己租用自行车，且不会借钱给他人，他们想知道最多有多少位同学能够租用到自行车。

本题采用二分法。对于区间 $[l,r]$ ，我们取中间点 `mid` 并判断租用到自行车的人数能否达到 `mid`。判断的过程是利用贪心算法实现的。

```
01 #include <iostream>
02 using namespace std;
03 #define MAXN 1000000
04 int n, B, A, M[MAXN], C[MAXN], l, r, ans, mid;
05 bool check(int nn) {
06     int count = 0, i, j;
07     i = __(1)__;
08     j = 1;
09     while (i <= n) {
10         if(__(2)__)
11             count += C[j] - M[i];
12         i++;
13         j++;
14     }
15     return __(3)__;
16 }
```

```
17 void sort(int a[], int l, int r) {
18     int i = l, j = r, x = a[(l + r) / 2], y;
19     while (i <= j) {
20         while (a[i] < x) i++;
21         while (a[j] > x) j--;
22         if (i <= j) {
23             y = a[i]; a[i] = a[j]; a[j] = y;
24             i++; j--;
25         }
26     }
27     if (i < r) sort(a, i, r);
28     if (l < j) sort(a, l, j);
29 }
30 int main() {
31     int i;
32     cin >> n >> B >> A;
33     for (i = 1; i <= n; i++)
34         cin >> M[i];
35     for (i = 1; i <= B; i++)
36         cin >> C[i];
37     sort(M, 1, n);
38     sort(C, 1, B);
39     l = 0;
40     r = n;
41     while (l <= r) {
42         mid = (l + r) / 2;
43         if(__(4)__){
44             ans = mid;
45             l = mid + 1;
46         }else
47             r = __(5)__;
48     }
49     cout << ans << endl;
50     return 0;
51 }
```

34. (3分) (1)处应填：_____

35. (3分) (2)处应填：_____

36. (3分) (3)处应填：_____

37. (2.5分) (4)处应填：_____

38. (2.5分) (5)处应填：_____

(2)

(马走日)回溯算法实际上一个类似枚举的搜索尝试过程，主要是在搜索尝试过程中寻找问题的解，当发现已不满足求解条件时，就“回溯”返回，尝试其他路径。回溯法是一种选优搜索法按选向前乘以达到目标。但当搜索到某一步时，若发现原先选择并优或达不到目标，则返回一步重新选择，这种是不通就返回去再现的技术称为回溯法，而满足回溯条件的某个状态的点称为回溯点。

马在中国棋棋中以日字形规则移动。请编写一段程序，给定n×大小的棋盘以及马的初始位置(m,n)，要求不能重复盘上的同一个点计算马有多少途迹还可以逾历盘上的所有点。

```
01 #include<iostream>
02 using namespace std;
03 int r,c;
04 int cnt,tot;
05 int wayr[8]={2,2,1,-1,-2,-2,1,-1};
06 int wayc[8]={1,-1,2,2,1,-1,-2,-2};
07 bool mark[1001][1001];
08 bool check(int x, int y)
09 {
10     if(__(1)__) return true;
11     return false;
12 }
13 void search(int x,int y)
14 {
15     for(int i=0;i<8;i++)
16         if(__(2)__ && __(3)__)
17         {
18             mark[x+wayr[i]][y+wayc[i]]=true;
19             tot++;
20             if(__(4)__) cnt++;
21             search(__(5)__);
22             tot--;
23             mark[x+wayr[i]][y+wayc[i]]=false;
24         }
25 }
26 int main()
27 {
28     cnt=0;
29     int m,n;
```

```
30      cin>>r>>c>>m>>n;
31      if(!check(m,n)) cout<<0<<endl;
32      else if(r==1&&c==1) cout<<1<<endl;
33      else
34      {
35          mark[m][n]=true;
36          search(m,n);
37          cout<<cnt<<endl;
38      }
39  }
```

39. (3分) (1)处应填

A. x>=0 || y>=0 || x<y < c . B. $x \geqslant 0 \mid y \geqslant 0 \&\& x < r \mid y < e.$

C. x>=0 && y>=0 && x<r && y<c D. x>=0&&y>=0 || x<r &&y<c

40. (3分) (2)处应填

A. check(wayr[i], wayc[i]) B. check(x+wayr[i] , y+wayc[i])

C. !check(x+wayr[i],y+wayc[i]) D. !check(wayr[l],wayc[i])

41. (3分) (3)处应填

A. mark[wayr[i]][wayc[i]] B. mark[x+wayr[i]][y+wayc[ii]]

C. !mark[wayr[i]][wayc[i]) D. !mark[x+wayr[i]][y+wayc[i]]

42. (3分) (4)处应填

A. tot==r * c-1 B. tot-=r * c C. cnt==r * c-1 D. cnt==r * c

43. (3分) (5)处应填

A. wayr[i], wayc[i] B. x-wayr[i], y-wayc[i]

C. x-wayr[i], y+wayc[i] D. x+wayr[i], y+wayc[i]