2023 年学而思网校 CSP-J 入门级初赛模拟题

（时间 120 分钟 满分 100 分）

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．以下哪个选项不是面向对象程序设计的特征（ ）。

A．封装

B．传承

C．继承

D．多态

2．关于栈和队列说法正确的是（ ）

A．队列的操作方式是先进后出

B．栈的操作方式是先进先出

C．队列只能在队尾插入和在对头删除元素

D．栈只能在栈底删除元素

3．某数列有 10000 个各不相同的元素，由低至高按序排列；现要对该数列进行二分法查找 (binary search),在最坏的情况下，需查找多少次：（ ）。

A．12

B．13

C．14

D．15

4．如下图，有一个无穷大的的栈 S ,在栈的右边排列着 1,2,3,4,5 共五个车厢。其中每个车厢可以向左行走，也可以进入栈 S 让后面的车厢通过。现已知第一个到达出口的是 4 号车厢，请问所有可能的到达出口的车厢排列总数。（ ）。

A．4

B．5

C．6

D．7

5．运行以下代码片段 cout 语句输出的结果是（ ）。

```
char a[10]="sdjahxgki",*p;
p=a+7;
cout<<*p<<endl;
```

A．x

B．g

C．k

D．i

6．已知某表达式的前缀形式为 -*+3456 ,则其对应的后缀表达式为（ ）,其中 +、-、*是运算符。

A．34+5*6-

B．3456+*-

C．3+4*5-6

D．345*+6-

7．请问将 5 个黑球和 3 个白球排成一行，不同的排法数量为（ ）。

A．10

B．15

C．56

D．336

8．无向图顶点的度为与该顶点相连的边的个数。无向图 G 有 18 条边，有 4 个 4 度顶点、2 个 3 度顶点，其余顶点的度均小于 3,则 G 至少有（　　）个顶点。

A．12

B．13

C．14

D．15

9．满二叉树的叶节点个数为 N，则它的节点总数为（　　）。

A．2*N-1

B．2*N+1

C．2^n -1

D．2^n+1

10．由 1 个 a，3 个 b 和 2 个 c 构成的所有字符串中，包含子串"abc"的共有（　　）个。

A．8

B．12

C．16

D．24

11．设某二叉树的前序遍历序列为 ACBGDEFH,其中序遍历序列为 GBCDAFEH,则其后序遍历序列为（　　）。

A．GBDCFHEA

B．GBCDFHEA

C．GBDCHFEA

D．GDCFHEBA

12．十六进制数 12.3 对应的十进制数是（　　）。

A．18.1875

B．18.0375

C．16.1875

D．16.0375

13．在待排序的数据表已经为有序时，下列排序算法中花费时间反而多的是（　　）。

A．冒泡排序算法

B．选择排序算法

C．插入排序算法

D．快速排序算法

14．给定一个字符集 {a,b,c,d,e,f} ,它们的出现频率分别是 {6,3,8,2,10,4}。下面哪个是相应的哈夫曼编码的正确集合：（　　）

A．00,1011,01,1010,11,100

B．00,100,110,000,0010,01

C．10,1011,11,0011,00,010

D．0011,10,11,0010,01,000

15．考虑以下 C++函数的定义（　　）。

```
int F(int n)
{
    if(n  <= 1) return  2;
    else return 2*F(n-2)+F(n-1);
}
```

请问，执行函数 F(5)的时候，返回值是（　　）。

A．6

B．10

C．22

D．42

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 1.5 分，选择题 3 分，共计 40 分）

（1）

```
1   #include <iostream>
2   using namespace std;
3   string s,  t;
4   char c;
5   int a;
6   int main() {
7       cin  >>  s;
8       for (int i  =  0;  i  <  s.size();  i++)
9       {
10          if ('A'  <=  s[i] && s[i]  <=  'Z')
11              a  =  (s[i]  -  'A'  +  25)  %  26;
12          else a  =  (s[i]  -  'a'  +  1)  %  26;
13          if (i  &  2)c  =  a  +  'a';
14          else c  =  a  +  'A';
15          t  =  t  +  c;
16      }
17      cout  <<  t  <<  endl;
18      return 0;
19  }
```

16. 删掉第 11 行的 ‘A'<=s[i]，输出结果会出现字母以外的字符。（　　）

17. 如果 s 的长度是偶数，那么输出的字符串中大写字母数量和小写字母的数量相同（　　）

18. 如果输入的 s="abcdEFGH"，输出的结果是（　　）

A. ZAbcFGhi

B. zaBCfgHI

C. BCdeDEfg

D. bcDEdeFG

19. 将第 13 行的 i&2 替换为下列哪条语句不会影响最终的答案?（　　）

A. i%4%2==0

B. i%4%2==1

C. i%2==1

D. i/2%2==1

20. 若输出的字符串 t=“AZabGPds”，那么输入的 s 不可能是（　　）。

A. BABCfocr

B. zABaHoEr

C. zABCHQCT

D. zyzCfQcr

（2）

```
1   #include <iostream>
2   using namespace std;
3   int n,m,p[10000],q[10000],a[10000];
4   bool check(int b){
5       int num = 0, len = m;
6       for (int i = 1; i <= m; i++) {
7           num = num + q[i] * b;
8           a[i] = num % 10;
9           num /= 10;
10      }
11      while (num > 0) {
12          a[++len] = num % 10;
13          num /= 10;
14      }
15      if (len != n) return len < n;
16      for (int i = n; i >= 1; i--)
17          if(a[i]<p[i]) return true;
18          else return false;
19      return true;
20  }
21  int main()
22  {
23      string s,t;
24      cin>>s>>t;
25      n=s.size();
26      m=t.size();
27      for (int i = 0; i < n; i++) p[n - i] = s[i] - '0';
28      for (int i = 0; i < m; i++) q[m - i] = t[i] - '0';
29      int l = 0, r = 10000000;
30      while (l < r) {
31          int mid = (l + r + 1) / 2;
32          if (check(mid))
33              l = mid;
34          else r=mid-1;
35      }
36      cout<<l<<endl;
37      return 0;
38  }
```

21. 如果 n 等于 m，输出一定是个位数。（　）
22. 如果 r-m 大于 7，第 34 行一定不会执行。（　）
23. 如果答案是 150，那么把 r 的初始值设置为 10^9，那么得到的答案还是 150。（ ）
24. 若输入的数字是 “620124” ,那么程序的输出为。（　）

    A. 5

    B. 6

    C. 7

    D. 8

25. 如果输入的数字是 “4245353461 5353242390” ,那么程序的第（　）行不会被执行。

    A. 14

    B. 20

    C. 34

    D. 35

26. 如果输入的数字是“3645353 1215117” ,那么第 8 行的运行次数最接近（　）次。

    A. 21

    B. 84

    C. 161

    D. 23

（3）

```
#include <iostream>
#include <string>
using namespace std;
string s;
int n, b[100], a[100], L[100], R[100], ans;
bool vis[100];
void dfs(int x, int dep)
{
    if (L[x]) dfs(L[x], dep + 1);
    if (R[x]) dfs(R[x], dep + 1);
         ans = ans + a[x] * dep;
}
int main()
{
    cin >> s;
    for (int i = 0; i < s.size(); i++)b[s[i] - 'A']++;
    for (int i = 0; i < 26; i++)
        if (b[i] > 0) a[++n] = b[i];
    a[0] = 1000000000;
    while (true) {
        int x = 0, y = 0;
        for (int j = 1; j <= n; j++)
            if (!vis[j]) {
                if (a[x] > a[j]) {
                    y = x;
                    x = j;
                }
                else if (a[y] > a[j])y = j;
            }
        if (y == 0)break;
        vis[x] = 1;vis[y] = 1;
        n++;
        L[n] = x;R[n] = y;
        a[n] = a[x] + a[y];
    }
    dfs(n, 0);
    cout << ans << endl;
    return 0;
}
```

27. 如果 s 仅由大写字母 A 组成，那么程序的输出定是定值。（　　）
28. 如果输入长度不同的字符串，得到的输出结果一定不同。（　　）
29. 把第 23 行的循环改成倒序，程序的输出结果可能发生改变。（　　）
30. 输入 s=“BCAAFFDDDFFCCACACCEE”,程序的输出是（　　）

A. 89

B. 90

C. 93

D. 159

31. 如果字符串 s 由 20 种大写字母构成，那么第 24 行的运行次数是（　　）次。

A. 210

B. 400

C. 551

D. 590

32. 如果字符串 s 中 26 种大写字母分别出现了 k 次，那么最终 ans 的值是（　　）。

A. 360k

B. 380k

C. 390k

D. 510k

三、 完善程序

（1） 水仙花数：

输入一个正整数 n($1 \le n \le 10^9$)，输出所有不超过 n 的水仙花数，并按照格式输出每个水仙花数展开的等式，每行一个。

若三位数 ABC 满足$ABC = A^3 + B^3 + C^3$，则称 ABC 为水仙花数，例如$153 = 1^3 + 5^3 + 3^3$，所以 153 是水仙花数。程序需要按照$153 = 1^3 + 5^3 + 3^3$的格式从高到低输出所有水仙花数，补全程序。

```
#include <iostream>
#include <cmath>
using namespace std;
int Cube(int x) {
    return ①;
}
int main() {
    int n;
    cin >> n;
    for (int i = ②; i <= n && i <= 999; ++i) {
        int a = ③;
        int b = i / 10 % 10;
        int c = ④;
        if (i == Cube(a) + Cube(b) + Cube(c)) {
            cout << i << "=" << a << ⑤;
        }
    }
    return 0;
}
```

33. ①处应填（　）。
    A. x^3
    B. pow(3,x)
    C. Cube(x-1)*x
    D. x*x*x
34. ②处应填（　）。
    A. 1
    B. 100
    C. sqrt(n)
    D. 0
35. ③处应填（　）。
    A. i%100
    B. i%(100%10)
    C. i/100
    D. i%10
36. ④处应填（　）。
    A. i-i/100*100-i/10*10
    B. i/10
    C. i/100%10
    D. i%10
37. ⑤处应填（　）。
    A. <<b<<"^3+"<<c<<"^3"<<endl
    B. "^3+"<<b<<"^3+"<<c<<"^3+"<<endl
    C. "^3+"<< b <<"^3+"<< c <<"^3\n"
    D. "^3+"<<b<<"^3+"<<c<< endl

（2）

(数独)现有一个 9*9 大小的二维数组，其中非 0 的数字表示该位置已填写的数，为 0 的数字 表示该位置尚未填写。现在要在所有为 0 的位置中填写数字，使得满足以下条件，若解存在，只 需找到满足条件的一组解即可：

每行 1~9 恰好各出现一次。

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

const int MAXN = 9;

bool row_state[MAXN][10];
bool col_state[MAXN][10];
bool square_state[MAXN][10];

int square_id(int r, int c) {
    return ①;
}

void change_state(int sudoku[MAXN][MAXN], int r, int c) {
    int value = sudoku[r][c];
    row_state[r][value] ^= 1;
    col_state[c][value] ^= 1;
    square_state[square_id(r, c)][value] ^= 1;
}

bool fill(int sudoku[MAXN][MAXN], int r, int c) {
    if (r >= MAXN)
        return true;

    ②;
    if (c == MAXN - 1) {
        next_r = r + 1;
        next_c = 0;
    }

    if (sudoku[r][c])
        ③;

    for (int num = 1; num <= 9; ++num) {
        if (④)
            continue;
        sudoku[r][c] = num;
        change_state(sudoku, r, c);
        if (fill(sudoku, next_r, next_c))
            return true;
        change_state(sudoku, r, c);
        ⑤;
    }

    return false;
};

int main() {
    int sudoku[MAXN][MAXN] = {{2, 0, 0, 0, 8, 0, 3, 0, 0},
        {0, 6, 0, 0, 7, 0, 0, 8, 4},
        {0, 3, 0, 5, 0, 0, 2, 0, 9},
        {0, 0, 0, 1, 0, 5, 4, 0, 8}
        {0, 0, 0, 0, 0, 0, 0, 0, 0},
        {4, 0, 2, 7, 0, 6, 0, 0, 0},
        {3, 0, 1, 0, 0, 7, 0, 4, 0},
        {7, 2, 0, 0, 4, 0, 0, 6, 0},
        {0, 0, 4, 0, 1, 0, 0, 0, 3}};

    for (int i = 0; i < MAXN; ++i)
        for (int j = 0; j < MAXN; ++j) {
            if (!sudoku[i][j])
                continue;
            change_state(sudoku, i, j);
        }

    if (!fill(sudoku, 0, 0)) {
        cout << "No Solution!" << endl;
        return 0;
    }

    for (inti = 0; i < MAXN; ++i) {
        for (int j = 0; j < MAXN; ++j)
            cout << sudoku[i][j] << ' ';
        cout << endl;
    }

```

```
    //输出:
    //2 4 5 9 8 1 3 7 6
    //1 6 9 2 7 3 5 8 4
    //8 3 7 5 6 4 2 1 9
    //9 7 6 1 2 5 4 3 8
    //5 1 3 4 9 8 6 2 7
    //4 8 2 7 3 6 9 5 1
    //3 9 1 6 5 7 8 4 2
    //7 2 8 3 4 9 1 6 5
    //6 5 4 8 1 2 7 9 3

    return 0;
}
```

38. ①处应填（ ）
    A. (r-1)/3*3+c/3+1
    B. r/3*3+c/3
    C. r/3*3+c%3
    D. r%3+c/3
39. ②处应填（ ）
    A. int next_r = r,next_c = c
    B. int next_r = r+1,next_c = c
    C. int next_r =r,next_c = c+1
    D. int next_r = r+1,next_c = c+1
40. ③处应填（ ）
    A. fill(sudoku,r,c)
    B. fill(sudoku,next_r,next_c)
    C. return fill(sudoku,r,c)
    D. return fil(sudoku,next_r,next_c)
41. ④处应填（ ）
    A. row_state[r][num]||col_state[c][num]||square_state[square_id(r,c)][num]
    B. !row_state[r][num]&&!col_state[c][num]&&!square_state[square_id(r,c)][num]
    C. row_state[r][c]&&col_state[r][c]&&square_state[r][c]
    D. !row_state[r][c]||!col_state[r][c]||!square_state[r][c]
42. ⑤处应填（ ）
    A. sudoku[r][c]^=1
    B. sudoku[r][c]=0
    C. row_state[r][num]=col_state[c][num]=square_state[square_id(r,c)][num]=false
    D. row_state[r][num]=col_state[c][num]=square_state[square_id(r,c)][num]=true