CSP-J 模拟（七）

2025年8月30日

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. 如果开始时计算机处于小写输入状态，现在有一只小老鼠反复按照 $CapsLock$、字母键 A、字母键 S 和字母键 D 的顺序循环按键，即

- $CapsLock$、A、S、D、$CapsLock$、A、S、D、……，

则屏幕上输出的第 2025 个字符是字母（　）。

A. A
B. S
C. D
D. a

2. 在 8 位二进制补码中，10101011 表示的是十进制下的（　）。

A. -85
B. -43
C. 43
D. 85

3. 给定一个含 N 个不相同数字的数组，在最坏情况下，找出其中最大或最小的数，至少需要 $N-1$ 次比较操作。则最坏情况下，在该数组中同时找最大与 最小的数至少需要（　）次比较操作。

（$\lceil\ \rceil$ 表示向上取整，$\lfloor\ \rfloor$ 表示向下取整）。

A. $\lceil \frac{3N}{2} \rceil - 2$
B. $\lfloor \frac{3N}{2} \rfloor - 2$
C. $2N-2$
D. $2N-4$

4. 表达式 $a*(b+c)-d$ 的后缀表达形式为（　）。

A. $abcd*+-$
B. $abc+*d-$
C. $abc*+d-$
D. $-+*abcd$

5. 约定二叉树的根节点高度为 1。一棵结点数为 2025 的二叉树最少有（　）个叶子结点；一棵结点数为2025 的二叉树最小的高度值是（　）。

A. $1, 11$
B. $1, 12$
C. $2, 11$
D. $2, 12$

6. 向一个栈顶指针为 hs 的链式栈中插入一个指针 s 指向的结点时，应执行（　）。

A. $hs \rightarrow next = s;$

B. $s \rightarrow next = hs; hs = s;$

C. $s \rightarrow next = hs \rightarrow next; hs \rightarrow next = s;$

D. $s \rightarrow next = hs; hs = hs \rightarrow next$

7. 2-3 树是满足两个条件的树：

   - 所有叶结点到根的路径长度相同
   - 所有非叶子结点有两个或三个子结点

   如果一棵 2-3 树有 10 个叶结点，那么它可能有（　）个非叶结点。

A. 3

B. 4

C. 6

D. 8

8. 由四个没有区别的点构成的简单无向连通图的个数是（　）。

A. 6

B. 7

C. 8

D. 9

9. 设含有 10个元素的集合的全部子集数为 S，其中由 7 个元素组成的子集数为 T，则 $\frac{T}{S}$ 的值（　）。

A. $\frac{5}{32}$

B. $\frac{15}{128}$

C. $\frac{1}{8}$

D. $\frac{21}{128}$

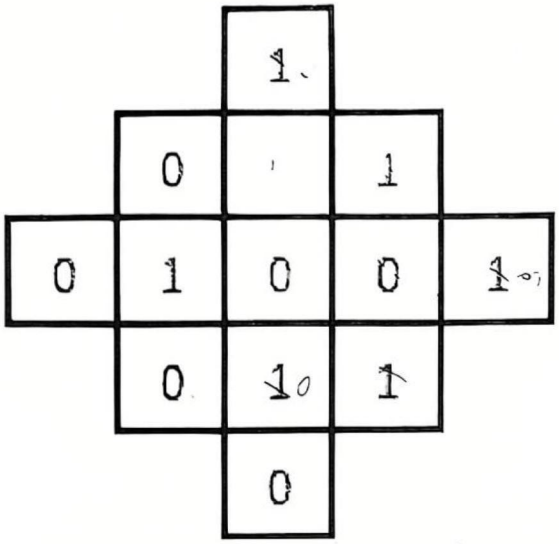
10. 如下图所示，共有 13 个格子。对任何一个格子进行一次操作，会使得它自己以及与它上下左右相邻的格子中的数字改变（由 1 变 0，或由 0 变 1）。现在要使得所有的格子中的数字都变为 0，至少需要（　）次操作。

A. 3

B. 4

C. 5

D. 6



11、一个 1×8的方格图形（不可旋转）用黑、白两种颜色填涂每个方格。如果每个方格只能填涂一种颜色，且不允许两个黑格相邻，共有（　）种填涂方案。

A. 8
B. 55
C. 56
D. 64

12. 设 G 是有 n 个结点、m 条边 $(n \leq m)$的连通图，必须删去G 的（　）条边，才能使得G 变成一棵树。

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

13. 有 7个一模一样的苹果，放到 3 个一样的盘子中，一共有（　）种放法。

A. 7
B. 8
C. 21
D. 2187

14. 若 $f_0=0, f_1=1, f_{n+1}=\dfrac{f_n+f_{n-1}}{2}$，则随着 i 的增大，f_i 将接近于（　　。

A. $\frac{1}{2}$
B. $\frac{2}{3}$
C. $\frac{\sqrt{5}-1}{2}$
D. 1

15. 以下是 32 位机器和 64 位机器的区别是（　）。

A. 显示器不同
B. 硬盘大小不同
C. 寻址空间不同
D. 输入法不同

二、阅读程序

- 判断题 1 分，选择题 3 分，共计 40 分
- 判断题正确填 T ，错误填 F ；

第1题

```
int solve1(int n)
{
  n++;
  int size = 0;
  int digit[16];
  int pow = 1;
  int s = 0;
  while (n > 0) {
    digit[size] = n % 10;
    size++;
    n /= 10;
    s += pow;
    pow *= 5;
  }
  while (size > 0) {
    --size;
    pow /= 5;
    s += (digit[size] / 2) * pow;
    if (digit[size] % 2 == 0) break;
  }
  return s - 1;
}

int solve2(int n)
{
    int s = 0;
    for(int i = 1; i <= n ; i += 2)
    {
        int t = i;
        bool pass = true;
        while (t > 0) {
            int x = t % 10;
            if(x%2 == 0)
            {
                pass = false;
                break;
            }
            t /= 10;
        }
        if (pass) s++;
    }
    return s;
}
```

保证 solve1 与 solve2 的参数 n 是非负整数。

判断题

16. solve1(11) 的返回值为 6 （ ）。

17. solve2(12345) 的返回值为 906 （ ）。

18. 对所有的 $n > 0$，必有 solve1(n) <= n （ ）。

19. solve1(0) 的返回值为 0 （ ）。

选择题

20. 关于 solve1(n) 与 solve2(n) 的大小关系，当 $n \geq 0$ 时，下列说法正确的是（ ）。

A. 必然有 solve1(n) < solve2(n)
B. 必然有 solve1(n) == solve2(n)
C. 只有当 n 为偶数时，才有 solve1(n) == solve2(n)
D. 只有当 n 为奇数时，才有 solve1(n) == solve2(n)

21. 若将 solve1 函数中第一句 n++ 去掉，将 solve2 函数中 for 循环的 i += 2 改为 i += 1，则新返回值与原来相比（ ）。

A. solve1 不变，solve2 变大
B. solve1 不变，solve2 变小
C. solve1 变小，solve2 不变
D. solve1 变大，solve2 不变

22. solve1(n) 与 solve2(n) 的时间复杂度为（ ）。

A. $\Theta(\log n)$、$\Theta(\log n)$
B. $\Theta(\log n)$、$\Theta(n)$
C. $\Theta(\log n)$、$\Theta(n \log n)$
D. $\Theta(n)$、$\Theta(n^2)$

第2题

```
int a[maxn];
int b[maxn];
int n, m;

const long long mod = 1'000'000'007;
bool filled[maxn][maxn];
long long mem[maxn][maxn];

long long solve(int i, int j)
{
  if (i == n) return 1;
  if (j == m) return 1;
  if (filled[i][j])
        return mem[i][j];
```

```
    filled[i][j] = true;

      long long sum = solve(i+1, j) + solve(i, j+1);
    if (a[i] == b[j]) {
      return mem[i][j] = sum % mod;
    }
    else {
      return mem[i][j] =  (sum - solve(i+1, j+1)) % mod;
    }
}

int main()
{
  std::cin >> n >> m;
  for (int i = 0; i < n; ++i) std::cin >> a[i];
  for (int i = 0; i < m; ++i) std::cin >> b[i];
  std::cout << (solve(0, 0) + mod) % mod << "\n";
}
```

判断题

23. 若依次输入数据 `2 2 1 3 3 1`，程序返回的结果是3（　　）。

24. 如果 `n = 0` 或 `m = 0`，程序返回的结果是1（　　）。

25. 如果两个序列完全相同，程序返回的结果是2^n。（　　）。

26. 输出时，执行 `solve(0, 0) + mod` 是多余的运算（　　）。

选择题

27. 该程序的主要功能是（　　）。

    A. 计算两个序列的最长公共子序列的长度
    B. 计算两个序列的公共子序列的数量
    C. 计算两个序列的公共子串的数量
    D. 计算两个序列的相同元素个数

28. 该程序的时间复杂度为（　　）。

    A. $\Theta(n+m)$
    B. $\Theta(n \cdot m)$
    C. $\Theta(2^{m+n})$
    D. $\Theta(n \log m)$

29. 当 `a[i] != b[j]` 时，返回值需要减去 `solve(i+1, j+1)` 的原因是（　　）。

    A. 排除重复情况
    B. 排除错误情况
    C. 优化程序的空间效率
    D. 优化程序的时间效率

第三题

```
long long k;
int n;
int p[20];
bool used[20] = {false};
long long frac[20];

void gen(int i)
{
  if (i > n)
    {
    for (int i = 1; i <= n; ++i) std::cout << p[i] << " ";
    return;
  }

  for (int a = 1; a <= n; ++a)
        if (not used[a]) {
      if (k <= frac[n-i]) {
        p[i] = a;
        used[a] = true;
        gen(i+1);
        return;
      }
      else {
        k -= frac[n-i];
      }
    }
}

int main()
{
    n = 1;
    frac[0] = frac[1] = 1;
  std::cin >> k;
  while (k > frac[n]) {
    k -= frac[n];
    frac[n+1] = frac[n] * (n+1);
    n++;
  }
  gen(1);
}
```

判断题

30. 当输入 k 为 1 时候，输出 1 （ ）。

31. 当输入 k 为 1000 时候，输出 1 5 2 4 3 （ ）。

选择题

32. 关于程序输出的序列所满足的性质，错误的是（　　）。

A. 输出的每个数字各不相同

B. 输出的数字都在 `1` 到 `n` 之间

C. 输出的相邻的数字差距不会超过 1

D. 最先输出的数字在程序中是最先被确定的。

33. 下列说法正确的是（　　）。

A. 越大的 k 一定会输出越长的序列

B. k 若是奇数，输出序列的长度一定也是奇数

C. 程序的时间复杂度为 $\Theta(k!)$

D. `gen` 递归是树形递归

34. 若程序输出的第一个数字为 `2`，第二数字为 `4`，剩下还有三个数字，则输入的 k（　　）。

A. 最小值是 60，最大值是 65

B. 最小值是 65，最大值是 70

C. 最小值是 70，最大值是 75

D. 最小值是 75，最大值是 80

35. 如果使得输出序列出现 7，则输入 k 最少需要（　　）。

A. 343

B. 874

C. 2170

D. 5040

三、完善程序

- 单选题，每小题 3 分，共计 30 分

第1题

有一个用户，在连续的 n 天里，都会收到积分，也会消费积分。积分在获得后的 m 天内有效（m 为一个给定的整数），过期失效。

在第 i 天，用户将会获得 p_i 分，他需要消费 c_i 分。若积分不足，则用掉全部积分后用其他方式消费。消费积分时，先用最早的。当天获取的积分可以当天消费。请计算这个用户一共消费了多少积分。

```
const int max_size = 100000;
int queue[max_size];
int head = 0;
int tail = 0;
int main()
{
    int n, m;
    std::cin >> n >> m;
    int sum = 0;
    for (int i = 0; i < n; ++i) {
        int p, c;
        std::cin >> p >> c;
        queue[____(1)____] = p;
        while ( ____(2)____ ) {
```

```
            if ( ____(3)____ ) {
                queue[head] -= c;
                sum += c;
                c = 0;
            }
            else {
                int amount = ____(4)____;
                c -= amount;
                sum += amount;
            }
        }
        if ( ____(5)____ > m) {
            head++;
        }
    }
    std::cout << sum << "\n";
}
```

36. (1) 处应填（ ）。

A. tail

B. tail+1

C. ++tail

D. tail++

37. (2)处应填（ ）。

A. head < tail || c > 0

B. head <= tail || c > 0

C. head < tail and c > 0

D. head <= tail and c > 0

38. (3)处应填（ ）。

A. queue[head] > c

B. queue[tail] > c

C. queue[head + 1] > c

D. queue[tail - 1] > c

39. (4)处应填（ ）。

A. queue[head++]

B. queue[++head]

C. queue[tail--]

D. queue[--tail]

40. (5)处应填（ ）。

A. head - tail

B. tail - head

C. tail

D. tail

第2题

给定一个 1 到 n 的排列 $p_1, p_2, \ldots, p_n$，请统计排列中所有长度大于等于 2 的连续子序列的次大数之和。

定义$\max_2(a_i, a_{i+1}, \ldots, a_j)$表示从$a_i$开始到$a_j$结束的连续子序列中，排名第二大的数，这个数就是一个连续子序列的次大数之和。

题目就是要求：

$\sum_{1 \le i < j \le n} \max_2(a_i, a_{i+1}, \ldots, a_j)$

solve 用于解决这个问题。

```
int q[maxn];
int prev[maxn];
int next[maxn];
long long solve(int n, int p[])
{
    for (int i = 1; i <= n; ++i)
    {
        ____(1)____ ;
    }
    p[0] = q[0] = prev[0] = 0;
    p[n+1] = q[n+1] = next[n+1] = n+1;
    for (int i = 1; i <= n; ++i)
    {
        int num = ____(2)____;
        int prev_num = p[i-1];
        int next_num = p[i+1];
        prev[num] = ____(3)____;
        next[num] = ____(4)____;
    }
    long long sum = 0;
    for (int num = 1; num <= n; ++num) {
        int prev_num = prev[num];
        int prev_prev_num = prev[prev_num];

        int next_num = next[num];
        int next_next_num = next[next_num];

        sum += (long long) num * ____(5)____ * (q[next_num] - q[num]);
        sum += (long long) num * (q[num] - q[prev_num]) * ____(6)____ ;

        ____(7)____ = prev_num;
        ____(8)____ = next_num;
    }
    return sum;
```

41. (1)处应填（ ）。

A. q[p[i]] = i

B. q[i] = p[i]

C. q[i] = 1

D. q[i] = i

42. (2) (3) (4)处应填（ ）。

A. p[i]， prev_num， next_num

B. q[i]， next_num， prev_num

C. p[i]， prev_num， next_num

D. q[i]， next_num， prev_num

43. (5) 处应填（ ）。

A. (q[prev_num] - q[prev_prev_num])

B. (q[next_next_num] - q[prev_num])

C. (q[next_num] - q[prev_prev_num])

D. (q[next_next_num] - q[next_num])

44. (6) 处应填（ ）。

A. (q[prev_num] - q[prev_prev_num])

B. (q[next_next_num] - q[prev_num])

C. (q[next_num] - q[prev_prev_num])

D. (q[next_next_num] - q[next_num])

45. (7)(8) 处应填（ ）。

A. prev[prev_num] , next[next_num]

B. prev[next_num] , next[prev_num]

C. next[prev_num] , prev[next_num]

D. next[next_num] , prev[prev_num]