0x22 综合卷复盘

做综合卷时，建议按正式考试节奏完成。完成后不要只看答案，还要复盘错误来源：是概念不清，还是边界没看准；是程序没有追踪完，还是选项没有逐项排除。

一、综合训练卷

（一）单项选择题

1. 在 C++ 中，下列哪个语句可以定义一个值不能通过变量名直接修改的整型变量？()

   A. `int x = 5;`
   B. `const int x = 5;`
   C. `unsigned int x = 5;`
   D. `static int x = 5;`

2. 二进制数 10110_2 与八进制数 35_8 的和，转换成十进制后为 ()。

   A. 49
   B. 50
   C. 51
   D. 57

3. 一个空栈中，元素按照 $1,2,3,4,5$ 的顺序依次入栈。下面哪一个序列不可能是出栈序列？()

   A. $1,3,2,5,4$
   B. $4,3,2,1,5$
   C. $2,4,3,5,1$
   D. $3,1,2,5,4$

4. 根结点高度为 1。一棵包含 100 个结点的完全二叉树，其高度至少为 ()。

   A. 5
   B. 6
   C. 7
   D. 8

5. 一棵完全二叉树用数组从下标 1 开始存储。若某结点存储在数组下标 11 的位置，则它的父结点下标和右孩子下标分别为 ()。

   A. $5, 22$
   B. $5, 23$

C. $6, 22$

D. $6, 23$

6. 后缀表达式 `5 1 2 + 4 * + 3 -` 的值为 ()。

A. 11

B. 12

C. 14

D. 18

7. 下列程序片段的时间复杂度最接近 ()。

```
int ans = 0;
for (int i = 1; i <= n; i *= 2) {
    for (int j = 1; j <= i; j++) {
        ans++;
    }
}
```

A. $O(n)$

B. $O(n \log n)$

C. $O(n^2)$

D. $O(\log n)$

8. 下列关于排序算法常见实现的说法，错误的是 ()。

A. 冒泡排序通常是稳定的

B. 插入排序通常是稳定的

C. 简单选择排序通常是稳定的

D. 归并排序可以实现为稳定排序

9. 从 8 名学生中选出 3 人参加活动，要求甲、乙两人中至少有一人被选中。不同的选择方式有 () 种。

A. 20

B. 28

C. 36

D. 42

10. 在一个有序数组中进行二分查找。若数组中有 100 个元素，则最坏情况下最多需要比较 () 次。

A. 6

B. 7

C. 8

D. 100

11. 有 4 个字符的出现频率分别为 $4, 5, 6, 7$。若使用哈夫曼编码，则对应哈夫曼树的带权路径长度为 ()。

A. 35
B. 39
C. 44
D. 52

12. 阅读以下递归函数，`f(5)` 的返回值为 ()。

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

A. 3
B. 5
C. 8
D. 13

13. 以下哪个不是操作系统？()

A. Linux
B. Windows
C. Android
D. HTML

14. 若定义 `int a[8];`，下列说法正确的是 ()。

A. 数组 `a` 中共有 8 个 `int` 类型元素
B. 数组 `a` 的合法下标为 1 到 8
C. `a[8]` 表示数组最后一个元素
D. 数组 `a` 中共有 9 个 `int` 类型元素

15. 在数据压缩中，哈夫曼编码方法在构造编码树时主要体现了哪种算法思想？()

A. 枚举
B. 贪心
C. 二分
D. 搜索

（二）阅读程序

程序 1

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

int f(int x) {
    int s = 0;
```

```
    while (x > 0) {
        s += x % 2;
        x /= 2;
    }
    return s;
}

int main() {
    int n;
    cin >> n;

    int cnt = 0, last = 0;
    for (int i = 1; i <= n; i++) {
        if (f(i) == 2) {
            cnt++;
            last = i;
        }
    }

    cout << cnt << " " << last << endl;
    return 0;
}
```

假设输入的 n 是不超过 1000 的正整数，完成下面的判断题和单选题。

判断题

16. `f(7)` 的返回值为 3。()
17. 当输入为 `5` 时，程序输出为 `2 5`。()
18. 该程序统计的是 1 到 n 中所有偶数的个数。()

单选题

19. 当输入为 `10` 时，程序输出为 ()。

    A. `4 10`
    B. `5 10`
    C. `5 9`
    D. `6 10`

20. 若将 `if (f(i) == 2)` 改为 `if (f(i) <= 2)`，当输入为 `5` 时，程序输出为 ()。

    A. `2 5`
    B. `3 5`
    C. `4 5`
    D. `5 5`

21. 该程序的时间复杂度最接近 ()。

    A. $O(n)$

B. $O(n \log n)$

C. $O(n^2)$

D. $O(2^n)$

程序 2

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

string h(string s) {
    string t;
    for (int i = 0; i < s.size(); i++) {
        if (s[i] == '#' && !t.empty()) {
            t.erase(t.size() - 1, 1);
        } else if (s[i] != '#') {
            t += s[i];
        }
    }
    return t;
}

int main() {
    string a, b;
    cin >> a >> b;

    string x = h(a);
    string y = h(b);

    cout << (x == y) << " " << x.size() + y.size() << endl;
    return 0;
}
```

假设输入字符串只由小写英文字母和字符 `#` 组成，且长度不超过 100。字符 `#` 表示删除当前结果的最后一个字符；若当前结果为空，则这个 `#` 不产生影响。完成下面的判断题和单选题。

判断题

22. 当输入为 `ab#c ac` 时，程序输出为 `1 4`。()

23. `h("#a")` 的返回值为 `#a`。()

24. `h("a##")` 的返回值为空字符串。()

单选题

25. 当输入为 `a#bc b#bc` 时，程序输出为 ()。

A. `1 4`

B. `0 4`

C. `1 2`

D. `0 2`

26. 当输入为 `abc#d abdd#` 时，程序输出为 ()。

A. `0 6`
B. `1 6`
C. `1 4`
D. `0 4`

27. 若将 `t.erase(t.size() - 1, 1);` 改为 `t += s[i];`，当输入为 `ab#c ac` 时，程序输出为 ()。

A. `1 4`
B. `0 4`
C. `0 6`
D. `1 6`

程序 3

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

const int N = 105;
int a[N], dp[N];

int main() {
    int n;
    cin >> n;
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }

    if (n == 1) {
        cout << a[1] << endl;
        return 0;
    }

    dp[1] = a[1];
    dp[2] = max(a[1], a[2]);

    for (int i = 3; i <= n; i++) {
        dp[i] = max(dp[i - 1], dp[i - 2] + a[i]);
    }

    cout << dp[n] << endl;
    return 0;
}
```

假设输入的 n 是不超过 100 的正整数，数组元素均为正整数。完成下面的判断题和单选题。

判断题

28. 当输入为 `4 1 2 3 1` 时，程序输出为 `4`。()
29. 当输入的 $n = 1$ 时，程序会直接输出 `a[1]` 的值。()
30. `dp[i]` 表示从前 i 个数中选出若干个互不相邻的数时，可以得到的最大和。()

单选题

31. 当输入为 `5 2 7 9 3 1` 时，程序输出为（）。

A. `11`
B. `12`
C. `13`
D. `22`

32. 若将 `dp[2] = max(a[1], a[2]);` 改为 `dp[2] = a[1] + a[2];`，当输入为 `4 5 1 1 5` 时，程序输出为（）。

A. `6`
B. `10`
C. `11`
D. `12`

33. 该程序的时间复杂度最接近（）。

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

（三）完善程序

程序 1（查找第一个不小于给定值的位置）

给定一个长度为 n 的升序数组 a 和一个整数 x，要求输出数组中第一个满足 $a_i \geq x$ 的位置。如果不存在这样的元素，输出 `-1`。数组下标从 1 开始。

试补全程序。

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

const int N = 1005;
int a[N];

int main() {
    int n, x;
    cin >> n >> x;
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }

    int l = 1, r = n, ans = -1;
    while (①) {
        int mid = ②;
        if (③) {
            ans = ④;
```

```
            r = mid - 1;
        } else {
            ⑤;
        }
    }

    cout << ans << endl;
    return 0;
}
```

34. ①处应填（）

A. `l <= r`
B. `l < 0`
C. `r < l`
D. `ans != -1`

35. ②处应填（）

A. `l + r`
B. `(l + r) / 2`
C. `r - l`
D. `(r - l) * 2`

36. ③处应填（）

A. `a[mid] < x`
B. `a[mid] == 0`
C. `a[mid] >= x`
D. `mid >= x`

37. ④处应填（）

A. `mid`
B. `a[mid]`
C. `x`
D. `l + r`

38. ⑤处应填（）

A. `r = mid - 1`
B. `l = mid - 1`
C. `r = mid + 1`
D. `l = mid + 1`

程序 2（网格路径计数）

给定一个 n 行 m 列的网格，`.` 表示可以经过的位置，`#` 表示障碍。现在从左上角 `(1,1)` 出发，每一步只能向下或向右走一格，不能经过障碍。求走到右下角 `(n,m)` 的不同路径条数。

试补全程序。

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

const int N = 35;
char mp[N][N];
long long dp[N][N];

int main() {
    int n, m;
    cin >> n >> m;

    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            cin >> mp[i][j];
            dp[i][j] = 0;
        }
    }

    if (①) {
        dp[1][1] = 1;
    } else {
        dp[1][1] = 0;
    }

    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            if (②) {
                dp[i][j] = 0;
                continue;
            }

            if (③) {
                continue;
            }

            if (i > 1) {
                dp[i][j] += ④;
            }
            if (j > 1) {
                dp[i][j] += ⑤;
            }
        }
    }

    cout << dp[n][m] << endl;
    return 0;
}
```

39. ①处应填（）

A. `mp[1][1] == '.'`

B. `mp[1][1] == '#'`

C. `n == m`

D. `dp[1][1] == 1`

40. ②处应填（）

A. `mp[i][j] == '#'`

B. `mp[i][j] == '.'`

C. `dp[i][j] > 0`

D. `i == n && j == m`

41. ③处应填（）

A. `i == 1 && j == 1`

B. `i == n && j == m`

C. `i > 1`

D. `j > 1`

42. ④处应填（）

A. `dp[i - 1][j]`

B. `dp[i][j - 1]`

C. `dp[i + 1][j]`

D. `dp[i][j + 1]`

43. ⑤处应填（）

A. `dp[i - 1][j]`

B. `dp[i][j - 1]`

C. `dp[i + 1][j]`

D. `dp[i][j + 1]`