CSP-J 2023 初赛真题

2023 CCF 非专业级别软件能力认证第一轮

(CSP-J1) 入门级 C++语言试题

认证时间：2023 年 9 月 16 日 09:30~11:30

考生注意事项：

- 试题纸共有 10 页，答题纸共有 1 页，满分 100 分。请在答题纸上作答，写在试题纸上的一律无效。
- 不得使用任何电子设备（如计算器、手机、电子词典等）或查阅任何书籍资料。

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. 在 C++中，下面哪个关键字用于声明一个变量，其值不能被修改？（ ）

A. unsigned

B. const

C. static

D. mutable

2. 八进制数12345670_8和07654321_8的和为（ ）。

A. 22222221_8

B. 21111111_8

C. 22111111_8

D. 22222211_8

3. 阅读下述代码，请问修改 data 的 value 成员以存储 3.14，正确的方式是（ ）。

```
union Data {
    int num;
    float value;
    char symbol;
};
union Data data;
```

A. data.value = 3.14;

B. value.data = 3.14;

C. data->value = 3.14;

D. value->data = 3.14;

4. 假设有一个链表的节点定义如下：

```
struct Node {
    int data;
    Node* next;
};
```

现在有一个指向链表头部的指针：Node* head。如果想要在链表中插入一个新节点，其成员 data 的值为 42，并使新节点成为链表的第一个节点，下面哪个操作是正确的？（ ）

A. Node* newNode = new Node; newNode->data = 42; newNode->next = head; head = newNode;

B. Node* newNode = new Node; head->data = 42; newNode->next = head; head = newNode;

C. Node* newNode = new Node; newNode->data = 42; head->next = newNode;

D. Node* newNode = new Node; newNode->data = 42; newNode->next = head;

5. 根节点的高度为 1，一棵拥有 2023 个节点的三叉树高度至少为（ ）。

A. 6

B. 7

C. 8

D. 9

6. 小明在某一天中依次有七个空闲时间段，他想要选出至少一个空闲时间段来练习唱歌，但他希望任意两个练习的时间段之间都有至少两个空闲的时间段让他休息。则小明一共有（ ）种选择时间段的方案。

A. 31

B. 18

C. 21

D. 33

7. 以下关于高精度运算的说法错误的是（ ）。

A. 高精度计算主要是用来处理大整数或需要保留多位小数的运算。

B. 大整数除以小整数的处理的步骤可以是，将被除数和除数对齐，从左到右逐位尝试将除数乘以某个数，通过减法得到新的被除数，并累加商。

C. 高精度乘法的运算时间只与参与运算的两个整数中长度较长者的位数有关。

D. 高精度加法运算的关键在于逐位相加并处理进位。

8. 后缀表达式“6 2 3 + - 3 8 2 / + * 2 ^ 3 +”对应的中缀表达式是（ ）。

A. ((6 - (2 + 3)) * (3 + 8 / 2)) ^ 2 + 3

B. 6 - 2 + 3 * 3 + 8 / 2 ^ 2 + 3

C. (6 - (2 + 3)) * ((3 + 8 / 2) ^ 2) + 3

D. 6 - ((2 + 3) * (3 + 8 / 2)) ^ 2 + 3

9. 数101010_2和166_8的和为（ ）。

A. 10110000_2

B. 236_8

C. 158_{10}

D. $A0_{16}$

10. 假设有一组字符{a,b,c,d,e,f}，对应的频率分别为 5%、9%、12%、13%、16%、45%，请问以下哪个选项是字符 a,b,c,d,e,f 分别对应的一组哈夫曼编码？（ ）

A. 1111, 1110, 101, 100, 110, 0

B. 1010, 1001, 1000, 011, 010, 00

C. 000, 001, 010, 011, 10, 11

D. 1010, 1011, 110, 111, 00, 01

11. 给定一棵二叉树，其前序遍历结果为：ABDECFG，中序遍历结果为：DEBACFG。请问这棵树的正确后序遍历结果是什么？（ ）

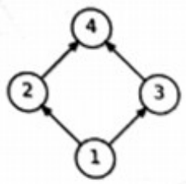
A. EDBFGCA

B. EDBGCFA

C. DEBGFCA

D. DBEGFCA

12. 考虑一个有向无环图，该图包含 4 条有向边：(1,2)，(1,3)，(2,4)和(3,4)。以下哪个选项是这个有向无环图的一个有效的拓扑排序？（ ）



A. 4, 2, 3, 1

B. 1, 2, 3, 4

C. 1, 2, 4, 3

D. 2, 1, 3, 4

13. 在计算机中，以下哪个选项描述的数据存储容量最小？（ ）

A. 字节（byte）

B. 比特（bit）

C. 字（word）

D. 千字节（kilobyte）

14. 一个班级有 10 个男生和 12 个女生。如果要选出一个 3 人的小组，并且小组中必须至少包含 1 个女生，那么有多少种可能的组合？（ ）

A. 1420

B. 1770

C. 1540

D. 2200

15. 以下哪个不是操作系统？（ ）

A. Linux

B. Windows

C. Android

D. HTML

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)

```
01 #include <iostream>
02 #include <cmath>
03 using namespace std;
04
05 double f(double a, double b, double c) {
06     double s = (a + b + c) / 2;
07     return sqrt(s * (s - a) * (s - b) * (s - c));
08 }
09
10 int main() {
11     cout.flags(ios::fixed);
12     cout.precision(4);
13
14     int a, b, c;
```

```
15      cin >> a >> b >> c;
16      cout << f(a, b, c) << endl;
17      return 0;
18  }
```

假设输入的所有数都为不超过 1000 的正整数，完成下面的判断题和单选题：

- 判断题

16.（2 分）当输入为“2 2 2”时，输出为“1.7321”。（ ）

17.（2 分）将第 7 行中的“(s - b) * (s - c)”改为“(s - c) * (s - b)”不会影响程序运行的结果。（ ）

18.（2 分）程序总是输出四位小数。（ ）

- 单选题

19. 当输入为“3 4 5”时，输出为（ ）。

A. “6.0000” B. “12.0000” C. “24.0000” D. “30.0000”

20. 当输入为“5 12 13”时，输出为（ ）。

A. “24.0000” B. “30.0000” C. “60.0000” D. “120.0000”

(2)

```
01 #include <iostream>
02 #include <vector>
03 #include <algorithm>
04 using namespace std;
05
06 int f(string x, string y) {
07     int m = x.size();
08     int n = y.size();
09     vector<vector<int>> v(m+1, vector<int>(n+1, 0));
10     for (int i = 1; i <= m; i++) {
11         for (int j = 1; j <= n; j++) {
12             if (x[i-1] == y[j-1]) {
13                 v[i][j] = v[i-1][j-1] + 1;
14             } else {
15                 v[i][j] = max(v[i-1][j], v[i][j-1]);
16             }
17         }
18     }
19     return v[m][n];
20 }
```



```
21
22 bool g(string x, string y) {
23     if (x.size() != y.size()) {
24         return false;
25     }
26     return f(x + x, y) == y.size();
27 }
28
29 int main() {
30     string x, y;
31     cin >> x >> y;
32     cout << g(x, y) << endl;
33     return 0;
34 }
```

- 判断题

21. f 函数的返回值小于等于 min(n,m)。（ ）

22. f 函数的返回值等于两个输入字符串的最长公共子串的长度。（ ）

23. 当输入两个完全相同的字符串时，g 函数的返回值总是 true。（ ）

- 单选题

24. 将第 19 行中的“v[m][n]”替换为“v[n][m]”，那么该程序（ ）。

A. 行为不变 B. 只会改变输出 C. 一定非正常退出 D. 可能非正常退出

25. 当输入为“csp-j p-jcs”时，输出为（ ）。

A. “0” B. “1” C. “T” D. “F”

26. 当输入为“csppsc spsccp”时，输出为（ ）。

A. “T” B. “F” C. “0” D. “1”

(3)

```
01 #include <iostream>
02 #include <cmath>
03 using namespace std;
04
05 int solve1(int n) {
06     return n * n;
07 }
08
09 int solve2(int n) {
```

```
10      int sum = 0;
11      for (int i = 1; i <= sqrt(n); i++) {
12          if (n % i == 0) {
13              if (n/i == i) {
14                  sum += i*i;
15              } else {
16                  sum += i*i + (n/i)*(n/i);
17              }
18          }
19      }
20      return sum;
21  }
22
23  int main() {
24      int n;
25      cin >> n;
26      cout << solve2(solve1(n)) << " " << solve1(solve2(n)) << endl;
27      return 0;
28  }
```

假设输入的 n 是绝对值不超过 1000 的整数，完成下面的判断题和单选题：

- 判断题

27. 如果输入的 n 为正整数，solve2 函数的作用是计算 n 所有的因子的平方和。（ ）

28. 第 13-14 行的作用是避免 n 的平方根因子 i（或 n/i）进入第 16 行而被计算两次。（ ）

29. 如果输入的 n 为质数，solve2(n)的返回值为n^2+1。（ ）

- 单选题

30. （4 分）如果输入的 n 为质数 p 的平方，那么 solve2(n)的返回值为（ ）。

A. p^2+p+1　　B. n^2+n+1　　C. n^2+1　　D. p^4+2p^2+1

31. 当输入为正整数时，第一项减去第二项的差值一定（ ）。

A. 大于 0　　B. 大于等于 0 且不一定大于 0　　C. 小于 0　　D. 小于等于 0 且不一定小于 0

32. 当输入为“5”时，输出为（ ）。

A. “651 625”　　B. “650 729”　　C. “651 676”　　D. “652 625”

三、完善程序（单选题，每小题 3 分，共计 30 分）

（1）（寻找被移除的元素）问题：原有长度为 n+1、公差为 1 的等差升序数列；将数列输入到程序的数组时移除了一个元素，导致长度为 n 的升序数组可能不再连续，除非被移除的是第一个或最后一个元素。需要在数组不连续时，找出被移除的元素。试补全程序。

```
01 #include <iostream>
02 #include <vector>
03
04 using namespace std;
05
06 int find_missing(vector<int>& nums) {
07     int left = 0, right = nums.size() - 1;
08     while (left < right) {
09         int mid = left + (right - left) / 2;
10         if (nums[mid] == mid + ①) {
11             ②;
12         } else {
13             ③;
14         }
15     }
16     return ④;
17 }
18
19 int main() {
20     int n;
21     cin >> n;
22     vector<int> nums(n);
23     for (int i = 0; i < n; i++) cin >> nums[i];
24     int missing_number = find_missing(nums);
25     if (missing_number == ⑤) {
26         cout << "Sequence is consecutive" << endl;
27     } else {
28         cout << "Missing number is " << missing_number << endl;
29     }
30     return 0;
31 }
```

33. ①处应填（ ）

A. 1　　B. nums[0]　　C. right　　D. left

34. ②处应填（ ）

A. left = mid + 1　　B. right = mid - 1
C. right = mid　　D. left = mid

35. ③处应填（ ）

A. left = mid + 1　　B. right = mid - 1
C. right = mid　　D. left = mid

36. ④处应填（ ）

A. left + nums[0]　　B. right + nums[0]
C. mid + nums[0]　　D. right + 1

37. ⑤处应填（ ）

A. nums[0]+n　　B. nums[0]+n-1　　C. nums[0]+n+1　　D. nums[n-1]

(2)(编辑距离)给定两个字符串，每次操作可以选择删除(Delete)、插入(Insert)、替换(Replace)一个字符，求将第一个字符串转换为第二个字符串所需要的最少操作次数。

试补全动态规划算法。

```
01 #include <iostream>
02 #include <string>
03 #include <vector>
04 using namespace std;
05
06 int min(int x, int y, int z) {
07     return min(min(x, y), z);
08 }
09
10 int edit_dist_dp(string str1, string str2) {
11     int m = str1.length();
12     int n = str2.length();
13     vector<vector<int>> dp(m + 1, vector<int>(n + 1));
14
15     for (int i = 0; i <= m; i++) {
16         for (int j = 0; j <= n; j++) {
17             if (i == 0)
18                 dp[i][j] = ①;
19             else if (j == 0)
20                 dp[i][j] = ②;
21             else if (③)
22                 dp[i][j] =④;
23             else
24                 dp[i][j] = 1 + min(dp[i][j - 1], dp[i - 1][j], ⑤);
25         }
26     }
27     return dp[m][n];
28 }
29
30 int main() {
31     string str1, str2;
32     cin >> str1 >> str2;
33     cout << "Minimum number of operations: "
34          << edit_dist_dp(str1, str2) << endl;
35   return 0;
36 }
```

38. ①处应填（ ）

A. j　　B. i　　C. m　　D. n

39. ②处应填（ ）

A. j　　B. i　　C. m　　D. n

40. ③处应填（ ）

A. str1[i - 1] == str2[j - 1]　　B. str1[i] == str2[j]
C. str1[i - 1] != str2[j - 1]　　D. str1[i] != str2[j]

41. ④处应填（ ）

A. dp[i - 1][j - 1] + 1　　B. dp[i - 1][j - 1]
C. dp[i - 1][j]　　D. dp[i][j - 1]

42. ⑤处应填（ ）

A. dp[i][j] + 1　　B. dp[i - 1][j - 1] + 1
C. dp[i - 1][j - 1]　　D. dp[i][j]

2023 CCF 非专业级别软件能力认证第一轮

（CSP-S1）提高级 C++语言试题

认证时间：2023 年 9 月 16 日 14:30~16:30

考生注意事项：

- 试题纸共有 13 页，答题纸共有 1 页，满分 100 分。请在答题纸上作答，写在试题纸上的一律无效。
- 不得使用任何电子设备（如计算器、手机、电子词典等）或查阅任何书籍资料。

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. 在 Linux 系统终端中，以下哪个命令用于创建一个新的目录？（ ）

A. newdir

B. mkdir

C. create

D. mkfolder

2. 0，1，2，3，4 中选取 4 个数字，能组成（ ）个不同四位数。（注：最小的四位数是 1000，最大的四位数是 9999。）

A. 96

B. 18

C. 120

D. 84

3. 假设 n 是图的顶点的个数，m 是图的边的个数，为求解某一问题有下面四种不同时间复杂度的算法。对于 $m = \Theta(n)$的稀疏图而言，下面的四个选项，哪一项的渐近时间复杂度最小。（ ）

A. $O(m\sqrt{\log n \cdot \log\log n})$

B. $O(n^2 + m)$

C. $O(n^2/\log m + m\log n)$

D. $O(m + n\log n)$

4. 假设有 n 根柱子，需要按照以下规则依次放置编号为 1、2、3、...的圆环：每根柱子的底部固定，顶部可以放入圆环；每次从柱子顶部放入圆环时，需要保证任何两个相邻圆环的编号之和是一个完全平方数。请计算当有 4 根柱子时，最多可以放置（ ）个圆环。

A. 7

B. 9

C. 11

D. 5

5. 以下对数据结构的表述不恰当的一项是：（ ）。

A. 队列是一种先进先出（FIFO）的线性结构

B. 哈夫曼树的构造过程主要是为了实现图的深度优先搜索

C. 散列表是一种通过散列函数将关键字映射到存储位置的数据结构

D. 二叉树是一种每个结点最多有两个子结点的树结构

6. 以下连通无向图中，（ ）一定可以用不超过两种颜色进行染色。

A. 完全三叉树

B. 平面图

C. 边双连通图

D. 欧拉图

7. 最长公共子序列长度常常用来衡量两个序列的相似度。其定义如下：给定两个序列 X = $\{x_1, x_2, x_3, \cdots, x_m\}$和 Y = $\{y_1, y_2, y_3, \cdots, y_n\}$，最长公共子序列（LCS）问题的目标是找到一个最长的新序列 Z = $\{z_1, z_2, z_3, \cdots, z_k\}$，使得序列 Z 既是序列 X 的子序列，又是序列 Y 的子序列，且序列 Z 的长度 k 在满足上述条件的序列里是最大的。（注：序列 A 是序列 B 的子序列，当且仅当在保持序列 B 元素顺序的情况下，从序列 B 中删除若干个元素，可以使得剩余的元素构成序列 A。）则序列“ABCAAAABA”和“ABABCBABA”的最长公共子序列长度为（ ）。

A. 4

B. 5

C. 6

D. 7

8. 一位玩家正在玩一个特殊的掷骰子的游戏，游戏要求连续掷两次骰子，收益规则如下：玩家第一次掷出 x 点，得到 2x 元；第二次掷出 y 点，当 y=x 时玩家会失去之前得到的 2x 元，而当 y≠x 时玩家能保住第一次获得的 2x 元。上述 x,y∈{1,2,3,4,5,6}。例如：玩家第一次掷出 3 点得到 6 元后，但第二次再次掷出 3 点，会失去之前得到的 6 元，玩家最终收

益为 0 元；如果玩家第一次掷出 3 点、第二次掷出 4 点，则最终收益是 6 元。假设骰子掷出任意一点的概率均为 1/6，玩家连续掷两次骰子后，所有可能情形下收益的平均值是多少？（ ）

A. 7 元

B. $\frac{35}{6}$元

C. $\frac{16}{3}$元

D. $\frac{19}{3}$元

9. 假设我们有以下的 C++代码：

```
int a = 5, b = 3, c = 4;
bool res = a & b || c ^ b && a | c;
```

请问，res 的值是什么？（ ）

提示：在 C++中，逻辑运算的优先级从高到低依次为：逻辑非（!）、逻辑与（&&）、逻辑或（||）。位运算的优先级从高到低依次为：位非（~）、位与（&）、位异或（^）、位或（|）。同时，双目位运算的优先级高于双目逻辑运算；逻辑非和位非优先级相同，且高于所有双目运算符。

A. true

B. false

C. 1

D. 0

10. 假设快速排序算法的输入是一个长度为 n 的已排序数组，且该快速排序算法在分治过程总是选择第一个元素作为基准元素。以下哪个选项描述的是在这种情况下的快速排序行为？（ ）

A. 快速排序对于此类输入的表现最好，因为数组已经排序。

B. 快速排序对于此类输入的时间复杂度是 $O(n \log n)$。

C. 快速排序对于此类输入的时间复杂度是 $O(n^2)$。

D. 快速排序无法对此类数组进行排序，因为数组已经排序。

11. 以下哪个命令，能将一个名为“main.cpp”的 C++源文件，编译并生成一个名为“main”的可执行文件？（ ）

A. g++ -o main main.cpp

B. g++ -o main.cpp main

C. g++ main -o main.cpp

D. g++ main.cpp -o main.cpp

12. 在图论中，树的重心是树上的一个结点，以该结点为根时，使得其所有的子树中结点数最多的子树的结点数最少。一棵树可能有多个重心。请问下面哪种树一定只有一个重心？（ ）。

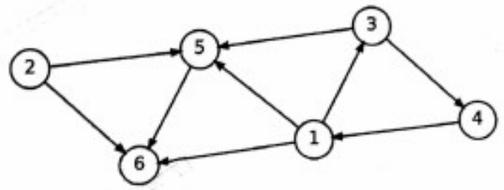
A. 4 个结点的树

B. 6 个结点的树

C. 7 个结点的树

D. 8 个结点的树

13. 如图是一张包含 6 个顶点的有向图，但顶点间不存在拓扑序。如果要删除其中一条边，使这 6 个顶点能进行拓扑排序，请问总共有多少条边可以作为候选的被删除边？（ ）



A. 1

B. 2

C. 3

D. 4

14. 若 $n = \sum_{i=0}^{k} 16^i \cdot x_i$，定义 $f(n) = \sum_{i=0}^{k} x_i$；其中 $x_i \in \{0, 1, \cdots, 15\}$。对于给定自然数 n_0，存在序列 $n_0, n_1, n_2, \cdots, n_m$，其中对于 $1 \le i \le m$ 都有 $n_i = f(n_{i-1})$，且 $n_m = n_{m-1}$，称 n_m 为 n_0 关于 f 的不动点。问在 100_{16} 至 $1A0_{16}$ 中，关于 f 的不动点为 9 的自然数个数为（ ）。

A. 10

B. 11

C. 12

D. 13

15. 现在用如下代码来计算x^n，其时间复杂度为（ ）。

```
double quick_power(double x, unsigned n) {
  if (n == 0) return 1;
  if (n == 1) return x;
  return quick_power(x, n / 2)
       * quick_power(x, n / 2)
       * ((n & 1) ? x : 1);
}
```

A. $O(n)$

B. $O(1)$

C. $O(\log n)$

D. $O(n \log n)$

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)

```
01 #include <iostream>
02 using namespace std;
03
04 unsigned short f(unsigned short x) {
05     x ^= x << 6;
06     x ^= x >> 8;
07     return x;
08 }
09
10 int main() {
11     unsigned short x;
12     cin >> x;
13     unsigned short y = f(x
14     cout << y <<endl;
15     return 0;
16 }
```

假设输入的 x 是不超过 65535 的自然数，完成下面的判断题和单选题：

- 判断题

16. 当输入非零时，输出一定不为零。（ ）

17. （2 分）将 f 函数的输入参数的类型改为 unsigned int，程序的输出不变。（ ）

18. 当输入为“65535”时，输出为“63”。（ ）

19. 当输入为“1”时，输出为“64”。（ ）

- 单选题

20. 当输入为“512”时，输出为（ ）。

A. “33280”　B. “33410”　C. “33106”　D. “33346”

21. 当输入为“64”时，执行完第 5 行后 x 的值为（ ）。

A. “8256”　B. “4130”　C. “4128”　D. “4160”

(2)

```
01 #include <iostream>
02 #include <cmath>
03 #include <vector>
04 #include <algorithm>
05 using namespace std;
06
07 long long solve1(int n) {
08     vector<bool> p(n+1, true);
09     vector<long long> f(n+1, 0), g(n+1, 0);
10     f[1] = 1;
11     for (int i = 2; i * i <= n; i++) {
12         if (p[i]) {
13             vector<int> d;
14             for (int k = i; k <= n; k *= i) d.push_back(k);
15             reverse(d.begin(), d.end());
16             for (int k : d) {
17                 for (int j = k; j <= n; j += k) {
18                     if (p[j]) {
19                         p[j] = false;
20                         f[j] = i;
21                         g[j] = k;
22                     }
23                 }
24             }
25         }
26     }
27     for (int i = sqrt(n) + 1; i <= n; i++) {
28         if (p[i]) {
```

```
29              f[i] = i;
30              g[i] = i;
31          }
32      }
33      long long sum = 1;
34      for (int i = 2; i <= n; i++) {
35          f[i] = f[i / g[i]] * (g[i] * f[i] - 1) / (f[i] - 1);
36          sum += f[i];
37      }
38      return sum;
39  }
40
41  long long solve2(int n) {
42      long long sum = 0;
43      for (int i = 1; i <= n; i++) {
44          sum += i * (n / i);
45      }
46      return sum;
47  }
48
49  int main() {
50      int n;
51      cin >> n;
52      cout << solve1(n) << endl;
53      cout << solve2(n) << endl;
54      return 0;
55  }
```

假设输入的 n 是不超过 1000000 的自然数，完成下面的判断题和单选题：

- 判断题

22. 将第 15 行删去，输出不变。（ ）

23. 当输入为“10”时，输出的第一行大于第二行。（ ）

24. （2 分）当输入为“1000”时，输出的第一行与第二行相等。（ ）

- 单选题

25. solve1(n) 的时间复杂度为（ ）。

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

26. solve2(n) 的时间复杂度为（ ）。

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

27. 输入为“5”时，输出的第二行为（ ）。

A. “20”　　B. “21”　　C. “22”　　D. “23”

(3)

```
01 #include <vector>
02 #include <algorithm>
03 #include <iostream>
04
05 using namespace std;
06
07 bool f0(vector<int>& a, int m, int k) {
08     int s = 0;
09     for (int i = 0, j = 0; i < a.size(); i++) {
10         while (a[i] - a[j] > m) j++;
11         s += i - j;
12     }
13     return s >= k;
14 }
15
16 int f(vector<int>& a, int k) {
17     sort(a.begin(), a.end());
18
19     int g = 0;
20     int h = a.back() - a[0];
21     while (g < h) {
22         int m = g + (h - g) / 2;
23         if (f0(a, m, k)) {
24             h = m;
25         } else {
26             g = m + 1;
27         }
28     }
29
30     return g;
31 }
32
33 int main() {
34     int n, k;
35     cin >> n >> k;
36     vector<int> a(n, 0);
37     for(int i = 0; i < n; i++) {
```

```
38          cin >> a[i];
39      }
40      cout<< f(a, k) << endl;
41      return 0;
42  }
```

假设输入总是合法的且$|a[i]|\le 10^8$、$n\le 10000$ 和 $1\le k\le n(n-1)/2$，完成下面的判断题和单选题：

- 判断题

28. 将第 24 行的“m”改为“m - 1”，输出有可能不变，而剩下情况为少 1。（ ）

29. 将第 22 行的“g + (h - g) / 2”改为“(h + g) >> 1”，输出不变。（ ）

30. 当输入为“5 7 2 -4 5 1 -3”，输出为“5”。（ ）

- 单选题

31. 设 a 数组中最大值减最小值加 1 为 A，则 f 函数的时间复杂度为（ ）。

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

32. 将第 10 行中的“>”替换为“>=”，那么原输出与现输出的大小关系为（ ）。

A. 一定小于　　B. 一定小于等于且不一定小于
C. 一定大于等于且不一定大于　　D. 以上三种情况都不对

33. 当输入为“5 8 2 -5 3 8 -12”时，输出为（ ）。

A. “13”　　B. “14”
C. “8”　　D. “15”

三、 完善程序（单选题，每小题 3 分，共计 30 分）

(1)（第 k 小路径）给定一张 n 个点 m 条边的有向无环图，顶点编号从 0 到 $n-1$。对于一条路径，我们定义“路径序列”为该路径从起点出发依次经过的顶点编号构成的序列。求所有至少包含一个点的简单路径中，“路径序列”字典序第 k 小的路径。保证存在至少 k 条路径。上述参数满足 $1\le n,m\le 10^5$ 和 $1\le k\le 10^{18}$。

在程序中，我们求出从每个点出发的路径数量。超过 10^{18} 的数都用 10^{18} 表示。然后我们根据 k 的值和每个顶点的路径数量，确定路径的起点，然后可以类似地依次求出路径中的每个点。

试补全程序。

```
01 #include <iostream>
02 #include <algorithm>
03 #include <vector>
04
05 const int MAXN = 100000;
06 const long long LIM = 1000000000000000000ll;
07
08 int n, m, deg[MAXN];
09 std::vector<int> E[MAXN];
10 long long k, f[MAXN];
11
12 int next(std::vector<int> cand, long long &k) {
13   std::sort(cand.begin(), cand.end());
14   for (int u : cand) {
15     if (①) return u;
16     k -= f[u];
17   }
18   return -1;
19 }
20
21 int main() {
22   std::cin >> n >> m >> k;
23   for (int i = 0; i < m; ++i) {
24     int u, v;
25     std::cin >> u >> v; // 一条从 u 到 v 的边
26     E[u].push_back(v);
27     ++deg[v];
28   }
29   std::vector<int> Q;
30   for (int i = 0; i < n; ++i)
31     if (!deg[i]) Q.push_back(i);
32   for (int i = 0; i < n; ++i) {
33     int u = Q[i];
34     for (int v : E[u]) {
35       if (②) Q.push_back(v);
36       --deg[v];
37     }
38   }
39   std::reverse(Q.begin(), Q.end());
40   for (int u : Q) {
41     f[u] = 1;
42     for (int v : E[u]) f[u] = ③;
43   }
44   int u = next(Q, k);
45   std::cout << u << std::endl;
```

```
46   while (④) {
47     ⑤;
48     u = next(E[u], k);
49     std::cout << u << std::endl;
50   }
51   return 0;
52 }
```

34. ①处应填（ ）

A. k >= f[u]　　B. k <= f[u]　　C. k > f[u]　　D. k < f[u]

35. ②处应填（ ）

A. deg[v] == 1　　B. deg[v] == 0　　C. deg[v] > 1　　D. deg[v] > 0

36. ③处应填（ ）

A. std::min(f[u] + f[v], LIM)　　B. std::min(f[u] + f[v] + 1, LIM)
C. std::min(f[u] * f[v], LIM)　　D. std::min(f[u] * (f[v] + 1), LIM)

37. ④处应填（ ）

A. u != -1　　B. !E[u].empty()　　C. k > 0　　D. k > 1

38. ⑤处应填（ ）

A. k += f[u]　　B. k -= f[u]　　C. --k　　D. ++k

(2)（最大值之和）给定整数序列 $a_0, \ldots, a_{n-1}$，求该序列所有**非空连续子序列**的最大值之和。上述参数满足 $1 \le n \le 10^5$ 和 $1 \le a_i \le 10^8$。

一个序列的非空连续子序列可以用两个下标 l 和 r（其中 $0 \le l \le r < n$）表示，对应的序列为 $a_l, a_{l+1}, \ldots, a_r$。两个非空连续子序列不同，当且仅当下标不同。

例如，当原序列为 [1,2,1,2] 时，要计算子序列 [1]、[2]、[1]、[2]、[1,2]、[2,1]、[1,2]、[1,2,1]、[2,1,2]、[1,2,1,2] 的最大值之和，答案为 18。注意 [1,1] 和 [2,2] 虽然是原序列的子序列，但不是连续子序列，所以不应该被计算。另外，注意其中有一些值相同的子序列，但由于他们在原序列中的下标不同，属于不同的非空连续子序列，所以会被分别计算。

解决该问题有许多算法，以下程序使用分治算法，时间复杂度 $O(n \log n)$。

试补全程序。

```
01 #include <iostream>
02 #include <algorithm>
03 #include <vector>
04
05 const int MAXN = 100000;
06
07 int n;
08 int a[MAXN];
09 long long ans;
10
11 void solve(int l, int r) {
12     if (l + 1 == r) {
13         ans += a[l];
14         return;
15     }
16     int mid = (l + r) >> 1;
17     std::vector<int> pre(a + mid, a + r);
18     for (int i = 1; i < r - mid; ++i) ①;
19     std::vector<long long> sum(r - mid + 1);
20     for (int i = 0; i < r - mid; ++i) sum[i + 1] = sum[i] + pre[i];
21     for (int i = mid - 1, j = mid, max = 0; i >= l; --i) {
22         while (j < r && ②) ++j;
23         max = std::max(max, a[i]);
24         ans += ③;
25         ans += ④;
26     }
27     solve(l, mid);
28     solve(mid, r);
29 }
30
31 int main() {
32     std::cin >> n;
33     for (int i = 0; i < n; ++i) std::cin >> a[i];
34     ⑤;
35     std::cout << ans << std::endl;
36     return 0;
37 }
```

39. ①处应填（ ）

A. pre[i] = std::max(pre[i - 1], a[i - 1])
B. pre[i + 1] = std::max(pre[i], pre[i + 1])
C. pre[i] = std::max(pre[i - 1], a[i])
D. pre[i] = std::max(pre[i], pre[i - 1])

40. ②处应填（ ）

A. a[j] < max　　B. a[j] < a[i]
C. pre[j - mid] < max　　D. pre[j - mid] > max

CS

41. ③处应填（ ）

A. (long long)(j - mid) * max
B. (long long)(j - mid) * (i - 1) * max
C. sum[j - mid]
D. sum[j - mid] * (i - 1)

42. ④处应填（ ）

A. (long long)(r - j) * max
B. (long long)(r - j) * (mid - i) * max
C. sum[r - mid] - sum[j - mid]
D. (sum[r - mid] - sum[j - mid]) * (mid - i)

43. ⑤处应填（ ）

A. solve(0, n)
B. solve(0, n - 1)
C. solve(1, n)
D. solve(1, n - 1)