

CSP2020-senior 试卷估分网站：<https://www.oitiku.com/>

**2020 CCF 非专业级别软件能力认证第一轮
(CSP-S) 提高级 C++语言试题**

认证时间：2020 年 10 月 11 日 09:30~11:30

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. 请选出以下最大的数（ ）。
A. $(550)_{10}$ B. $(777)_8$ C. 2^{10} D. $(22F)_{16}$
2. 操作系统的功能是（ ）。
A. 负责外设与主机之间的信息交换
B. 控制和管理计算机系统的各种硬件和软件资源的使用
C. 负责诊断机器的故障
D. 将源程序编译成目标程序
3. 现有一段 8 分钟的视频文件，它的播放速度是每秒 24 帧图像，每帧图像是一幅分辨率为 2048×1024 像素的 32 位真彩色图像。请问要存储这段原始无压缩视频，需要多大的存储空间？（ ）。
A. 30G B. 90G C. 150G D. 450G
4. 今有一空栈 S，对下列待进栈的数据元素序列 a, b, c, d, e, f 依次进行：进栈，进栈，出栈，进栈，进栈，出栈的操作，则此操作完成后，栈底元素为（ ）。
A. b B. a C. d D. c
5. 将 (2, 7, 10, 18) 分别存储到某个地址区间为 0~10 的哈希表中，如果哈希函数 $h(x) = ()$ ，将不会产生冲突，其中 $a \bmod b$ 表示 a 除以 b 的余数。
A. $x^2 \bmod 11$
B. $2x \bmod 11$
C. $x \bmod 11$
D. $\lfloor x/2 \rfloor \bmod 11$ ，其中 $\lfloor x/2 \rfloor$ 表示 $x/2$ 下取整
6. 下列哪些问题不能用贪心法精确求解？（ ）



- A. 霍夫曼编码问题 B. 0-1 背包问题
 C. 最小生成树问题 D. 单源最短路径问题
7. 具有 n 个顶点， e 条边的图采用邻接表存储结构，进行深度优先遍历运算的时间复杂度为（ ）。
 A. $\Theta(n+e)$ B. $\Theta(n^2)$ C. $\Theta(e^2)$ D. $\Theta(n)$
8. 二分图是指能将顶点划分成两个部分，每一部分内的顶点间没有边相连的简单无向图。那么，24 个顶点的二分图至多有（ ）条边。
 A. 144 B. 100 C. 48 D. 122
9. 广度优先搜索时，一定需要用到的数据结构是（ ）。
 A. 栈 B. 二叉树 C. 队列 D. 哈希表
10. 一个班学生分组做游戏，如果每组三人就多两人，每组五人就多三人，每组七人就多四人，问这个班的学生人数 n 在以下哪个区间？已知 $n < 60$ 。（ ）。
 A. $30 < n < 40$ B. $40 < n < 50$ C. $50 < n < 60$ D. $20 < n < 30$
11. 小明想通过走楼梯来锻炼身体，假设从第 1 层走到第 2 层消耗 10 卡热量，接着从第 2 层走到第 3 层消耗 20 卡热量，再从第 3 层走到第 4 层消耗 30 卡热量，依此类推，从第 k 层走到第 $k+1$ 层消耗 $10k$ 卡热量 ($k > 1$)。如果小明想从 1 层开始，通过连续向上爬楼梯消耗 1000 卡热量，至少要爬到第几层楼？（ ）。
 A. 14 B. 16 C. 15 D. 13
12. 表达式 $a * (b + c) - d$ 的后缀表达形式为（ ）。
 A. $abc * + d -$ B. $- + * abcd$ C. $abcd * + -$ D. $abc + * d -$
13. 从一个 4×4 的棋盘中选取不在同一行也不在同一列上的两个方格，共有（ ）种方法。
 A. 60 B. 72 C. 86 D. 64
14. 对一个 n 个顶点、 m 条边的带权有向简单图用 Dijkstra 算法计算单源最短路时，如果不使用堆或其它优先队列进行优化，则其时间复杂度为（ ）。
 A. $\Theta((m + n^2) \log n)$ B. $\Theta(mn + n^3)$
 C. $\Theta((m + n) \log n)$ D. $\Theta(n^2)$
15. 1948 年，（ ）将热力学中的熵引入信息通信领域，标志着信息论研究的开端。
 A. 欧拉 (Leonhard Euler) B. 冯·诺伊曼 (John von Neumann)
 C. 克劳德·香农 (Claude Shannon) D. 图灵 (Alan Turing)



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 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

- 1) n 必须小于 1000，否则程序可能会发生运行错误。 ()
- 2) 输出一定大于等于 0。 ()
- 3) 若将第 13 行的“ $j = 0$ ”改为“ $j = i + 1$ ”，程序输出可能会改变。
()
- 4) 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

- 5) 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 |
|--------|--------|--------|--------|
- 6) 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

2.

```

01 #include <iostream>
02 #include <cstdlib>
03 using namespace std;
04
05 int n;
06 int d[10000];
07
08 int find(int L, int R, int k) {
09     int x = rand() % (R - L + 1) + L;
10     swap(d[L], d[x]);
11     int a = L + 1, b = R;
12     while (a < b) {
13         while (a < b && d[a] < d[L])
14             ++a;
15         while (a < b && d[b] >= d[L])
16             --b;
17         swap(d[a], d[b]);
18     }
19     if (d[a] < d[L])
20         ++a;
21     if (a - L == k)
22         return d[L];
23     if (a - L < k)
24         return find(a, R, k - (a - L));
25     return find(L + 1, a - 1, k);
26 }
27
28 int main() {
29     int k;
30     cin >> n;
31     cin >> k;
32     for (int i = 0; i < n; ++i)
33         cin >> d[i];
34     cout << find(0, n - 1, k);
35     return 0;
36 }
```



假设输入的 n , k 和 $d[i]$ 都是不超过 10000 的正整数, 且 k 不超过 n , 并假设 `rand()` 函数产生的是均匀的随机数, 完成下面的判断题和单选题:

● 判断题

- 1) 第 9 行的 “`x`” 的数值范围是 $L+1$ 到 R , 即 $[L+1, R]$ 。 ()
- 2) 将第 19 行的 “`d[a]`” 改为 “`d[b]`”, 程序不会发生运行错误。 ()

● 单选题

- 3) (2.5 分) 当输入的 $d[i]$ 是严格单调递增序列时, 第 17 行的 “`swap`” 平均执行次数是 ()。
 - A. $\Theta(n \log n)$
 - B. $\Theta(n)$
 - C. $\Theta(\log n)$
 - D. $\Theta(n^2)$
- 4) (2.5 分) 当输入的 $d[i]$ 是严格单调递减序列时, 第 17 行的“`swap`” 平均执行次数是 ()。
 - A. $\Theta(n^2)$
 - B. $\Theta(n)$
 - C. $\Theta(n \log n)$
 - D. $\Theta(\log n)$
- 5) (2.5 分) 若输入的 $d[i]$ 为 i , 此程序①平均的时间复杂度和②最坏情况下的时间复杂度分别是 ()。
 - A. $\Theta(n), \Theta(n^2)$
 - B. $\Theta(n), \Theta(n \log n)$
 - C. $\Theta(n \log n), \Theta(n^2)$
 - D. $\Theta(n \log n), \Theta(n \log n)$
- 6) (2.5 分) 若输入的 $d[i]$ 都为同一个数, 此程序平均的时间复杂度是 ()。
 - A. $\Theta(n)$
 - B. $\Theta(\log n)$
 - C. $\Theta(n \log n)$
 - D. $\Theta(n^2)$

3.

```

01 #include <iostream>
02 #include <queue>
03 using namespace std;
04
05 const int maxl = 2000000000;
06
07 class Map {
08     struct item {
09         string key; int value;
10     } d[maxl];
11     int cnt;
12 public:
13     int find(string x) {
14         for (int i = 0; i < cnt; ++i)
15             if (d[i].key == x)
16                 return d[i].value;
17     return -1;

```



```

18    }
19    static int end() { return -1; }
20    void insert(string k, int v) {
21        d[cnt].key = k; d[cnt++].value = v;
22    }
23 } s[2];
24
25 class Queue {
26     string q[maxl];
27     int head, tail;
28 public:
29     void pop() { ++head; }
30     string front() { return q[head + 1]; }
31     bool empty() { return head == tail; }
32     void push(string x) { q[++tail] = x; }
33 } q[2];
34
35 string st0, st1;
36 int m;
37
38 string LtoR(string s, int L, int R) {
39     string t = s;
40     char tmp = t[L];
41     for (int i = L; i < R; ++i)
42         t[i] = t[i + 1];
43     t[R] = tmp;
44     return t;
45 }
46
47 string RtoL(string s, int L, int R) {
48     string t = s;
49     char tmp = t[R];
50     for (int i = R; i > L; --i)
51         t[i] = t[i - 1];
52     t[L] = tmp;
53     return t;
54 }
55
56 bool check(string st, int p, int step) {
57     if (s[p].find(st) != s[p].end())
58         return false;
59     ++step;
60     if (s[p ^ 1].find(st) == s[p].end()) {

```



```

61     s[p].insert(st, step);
62     q[p].push(st);
63     return false;
64 }
65 cout << s[p ^ 1].find(st) + step << endl;
66 return true;
67 }
68
69 int main() {
70     cin >> st0 >> st1;
71     int len = st0.length();
72     if (len != st1.length()) {
73         cout << -1 << endl;
74         return 0;
75     }
76     if (st0 == st1) {
77         cout << 0 << endl;
78         return 0;
79     }
80     cin >> m;
81     s[0].insert(st0, 0); s[1].insert(st1, 0);
82     q[0].push(st0); q[1].push(st1);
83     for (int p = 0;
84         !(q[0].empty() && q[1].empty());
85         p ^= 1) {
86         string st = q[p].front(); q[p].pop();
87         int step = s[p].find(st);
88         if ((p == 0 &&
89             (check(LtoR(st, m, len - 1), p, step) ||
90             check(RtoL(st, 0, m), p, step)))
91             ||
92             (p == 1 &&
93             (check(LtoR(st, 0, m), p, step) ||
94             check(RtoL(st, m, len - 1), p, step))))
95             return 0;
96     }
97     cout << -1 << endl;
98     return 0;
99 }

```

● 判断题

- 1) 输出可能为 0。 ()



2) 若输入的两个字符串长度均为 101 时，则 $m=0$ 时的输出与 $m=100$ 时的输出是一样的。 ()

3) 若两个字符串的长度均为 n ，则最坏情况下，此程序的时间复杂度为 $\Theta(n!)$ 。 ()

● 单选题

4) (2.5 分) 若输入的第一个字符串长度由 100 个不同的字符构成，第二个字符串是第一个字符串的倒序，输入的 m 为 0 ，则输出为 ()。

- A. 49 B. 50 C. 100 D. -1

5) (4 分) 已知当输入为 “0123\n3210\n1” 时输出为 4 ，当输入为 “012345\n543210\n1” 时输出为 14 ，当输入为 “01234567\n76543210\n1” 时输出为 28 ，则当输入为 “0123456789ab\nba9876543210\n1” 输出为 ()。其中 “\n” 为换行符。

- A. 56 B. 84 C. 102 D. 68

6) (4 分) 若两个字符串的长度均为 n ，且 $0 < m < n-1$ ，且两个字符串的构成相同（即任何一个字符在两个字符串中出现的次数均相同），则下列说法正确的是 ()。提示：考虑输入与输出有多少对字符前后顺序不一样。

- A. 若 n 、 m 均为奇数，则输出可能小于 0 。
 B. 若 n 、 m 均为偶数，则输出可能小于 0 。
 C. 若 n 为奇数、 m 为偶数，则输出可能小于 0 。
 D. 若 n 为偶数、 m 为奇数，则输出可能小于 0 。

三、完善程序（单选题，每小题 3 分，共计 30 分）

1. (分数背包) 小 S 有 n 块蛋糕，编号从 1 到 n 。第 i 块蛋糕的价值是 w_i ，体积是 v_i 。他有一个大小为 B 的盒子来装这些蛋糕，也就是说装入盒子的蛋糕的体积总和不能超过 B 。

他打算选择一些蛋糕装入盒子，他希望盒子里装的蛋糕的价值之和尽量大。

为了使盒子里的蛋糕价值之和更大，他可以任意切割蛋糕。具体来说，他可以选择一个 α ($0 < \alpha < 1$)，并将一块价值是 w ，体积为 v 的蛋糕切割成两块，其中一块的价值是 $\alpha \cdot w$ ，体积是 $\alpha \cdot v$ ，另一块的价值是 $(1 - \alpha) \cdot w$ ，体积是 $(1 - \alpha) \cdot v$ 。他可以重复无限次切割操作。

现要求编程输出最大可能的价值，以分数的形式输出。

比如 $n=3$, $B=8$, 三块蛋糕的价值分别是 4 、 4 、 2 ，体积分别是 5 、 3 、 2 。那么最优的方案就是将体积为 5 的蛋糕切成两份，一份体积是 3 ，价值是 2.4 ，另一份体积是 2 ，价值是 1.6 ，然后把体积是 3 的那部分和后两块蛋糕打包进盒子。最优的价值之和是 8.4 ，故程序输出 $42/5$ 。



输入的数据范围为： $1 \leq n \leq 1000$, $1 \leq B \leq 10^5$; $1 \leq w_i, v_i \leq 100$ 。
 提示：将所有的蛋糕按照性价比 w_i/v_i 从大到小排序后进行贪心选择。
 试补全程序。

```

01 #include <cstdio>
02 using namespace std;
03
04 const int maxn = 1005;
05
06 int n, B, w[maxn], v[maxn];
07
08 int gcd(int u, int v) {
09     if(v == 0)
10         return u;
11     return gcd(v, u % v);
12 }
13
14 void print(int w, int v) {
15     int d = gcd(w, v);
16     w = w / d;
17     v = v / d;
18     if(v == 1)
19         printf("%d\n", w);
20     else
21         printf("%d/%d\n", w, v);
22 }
23
24 void swap(int &x, int &y) {
25     int t = x; x = y; y = t;
26 }
27
28 int main() {
29     scanf("%d %d", &n, &B);
30     for(int i = 1; i <= n; i++) {
31         scanf("%d%d", &w[i], &v[i]);
32     }
33     for(int i = 1; i < n; i++)
34         for(int j = 1; j < n; j++)
35             if(①) {
36                 swap(w[j], w[j + 1]);
37                 swap(v[j], v[j + 1]);
38             }
39     int curV, curW;
```



```

40     if(②) {
41         ③
42     } else {
43         print(B * w[1], v[1]);
44         return 0;
45     }
46
47     for(int i = 2; i <= n; i++)
48     if(curV + v[i] <= B) {
49         curV += v[i];
50         curW += w[i];
51     } else {
52         print(④);
53         return 0;
54     }
55     print(⑤);
56     return 0;
57 }
58
59

```

1) ①处应填 ()

- A. $w[j] / v[j] < w[j + 1] / v[j + 1]$
 B. $w[j] / v[j] > w[j + 1] / v[j + 1]$
 C. $v[j] * w[j + 1] < v[j + 1] * w[j]$
 D. $w[j] * v[j + 1] < w[j + 1] * v[j]$

2) ②处应填 ()

- A. $w[1] \leq B$ B. $v[1] \leq B$ C. $w[1] \geq B$ D. $v[1] \geq B$

3) ③处应填 ()

- A. `print(v[1], w[1]); return 0;`
 B. `curV = 0; curW = 0;`
 C. `print(w[1], v[1]); return 0;`
 D. `curV = v[1]; curW = w[1];`

4) ④处应填 ()

- A. `curW * v[i] + curV * w[i], v[i]`
 B. `(curW - w[i]) * v[i] + (B - curV) * w[i], v[i]`
 C. `curW + v[i], w[i]`
 D. `curW * v[i] + (B - curV) * w[i], v[i]`

5) ⑤处应填 ()



- | | |
|---------------|------------|
| A. curW, curV | B. curW, 1 |
| C. curV, curW | D. curV, 1 |

2. (最优子序列) 取 $m = 16$, 给出长度为 n 的整数序列 $a_1, a_2, \dots, a_n (0 \leq a_i < 2^m)$ 。对于一个二进制数 x , 定义其分值 $w(x)$ 为 $x + \text{popcnt}(x)$, 其中 $\text{popcnt}(x)$ 表示 x 二进制表示中 1 的个数。对于一个子序列 b_1, b_2, \dots, b_k , 定义其子序列分值 S 为 $w(b_1 \oplus b_2) + w(b_2 \oplus b_3) + w(b_3 \oplus b_4) + \dots + w(b_{k-1} \oplus b_k)$ 。其中 \oplus 表示按位异或。对于空子序列, 规定其子序列分值为 0。求一个子序列使得其子序列分值最大, 输出这个最大值。

输入第一行包含一个整数 $n (1 \leq n \leq 40000)$ 。接下来一行包含 n 个整数 a_1, a_2, \dots, a_n 。

提示: 考虑优化朴素的动态规划算法, 将前 $\frac{m}{2}$ 位和后 $\frac{m}{2}$ 位分开计算。

$\text{Max}[x][y]$ 表示当前的子序列下一个位置的高 8 位是 x 、最后一个位置的低 8 位是 y 时的最大价值。

试补全程序。

```

01 #include <iostream>
02
03 using namespace std;
04
05 typedef long long LL;
06
07 const int MAXN = 40000, M = 16, B = M >> 1, MS = (1 <<
B) - 1;
08 const LL INF = 1000000000000000LL;
09 LL Max[MS + 4][MS + 4];
10
11 int w(int x)
12 {
13     int s = x;
14     while (x)
15     {
16         ①;
17         s++;
18     }
19     return s;
20 }
21
22 void to_max(LL &x, LL y)
23 {

```



```

24     if (x < y)
25         x = y;
26     }
27
28 int main()
29 {
30     int n;
31     LL ans = 0;
32     cin >> n;
33     for (int x = 0; x <= MS; x++)
34         for (int y = 0; y <= MS; y++)
35             Max[x][y] = -INF;
36     for (int i = 1; i <= n; i++)
37     {
38         LL a;
39         cin >> a;
40         int x = ②, y = a & MS;
41         LL v = ③;
42         for (int z = 0; z <= MS; z++)
43             to_max(v, ④);
44         for (int z = 0; z <= MS; z++)
45             ⑤;
46         to_max(ans, v);
47     }
48     cout << ans << endl;
49     return 0;
50 }

```

1) ①处应填 ()

- A. $x \gg= 1$
- B. $x \wedge= x \& (x \wedge (x + 1))$
- C. $x -= x \mid -x$
- D. $x \wedge= x \& (x \wedge (x - 1))$

2) ②处应填 ()

- | | |
|----------------------|----------------------|
| A. $(a \& MS) \ll B$ | B. $a \gg B$ |
| C. $a \& (1 \ll B)$ | D. $a \& (MS \ll B)$ |

3) ③处应填 ()

- | | |
|-----------|----------------|
| A. $-INF$ | B. $Max[y][x]$ |
| C. 0 | D. $Max[x][y]$ |

4) ④处应填 ()



- A. $\text{Max}[x][z] + w(y \wedge z)$ B. $\text{Max}[x][z] + w(a \wedge z)$
C. $\text{Max}[x][z] + w(x \wedge (z \ll B))$ D. $\text{Max}[x][z] + w(x \wedge z)$

5) ⑤处应填 ()

- A. `to_max(Max[y][z], v + w(a ^ (z << B)))`
B. `to_max(Max[z][y], v + w((x ^ z) << B))`
C. `to_max(Max[z][y], v + w(a ^ (z << B)))`
D. `to_max(Max[x][z], v + w(y ^ z))`

CSP2020-junior 试卷

**2020 CCF 非专业级别软件能力认证第一轮
(CSP-J) 入门级 C++语言试题**

认证时间：2020 年 10 月 11 日 14:30~16:30

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. 在内存储器中每个存储单元都被赋予一个唯一的序号，称为（ ）。
A. 下标 B. 地址 C. 序号 D. 编号
2. 编译器的主要功能是（ ）。
A. 将源程序翻译成机器指令代码
B. 将一种高级语言翻译成另一种高级语言
C. 将源程序重新组合
D. 将低级语言翻译成高级语言
3. 设 $x=true$, $y=true$, $z=false$, 以下逻辑运算表达式值为真的是（ ）。
A. $(x \wedge y) \wedge z$ B. $x \wedge (z \vee y) \wedge z$
C. $(x \wedge y) \vee (z \vee x)$ D. $(y \vee z) \wedge x \wedge z$
4. 现有一张分辨率为 2048×1024 像素的 32 位真彩色图像。请问要存储这张图像，需要多大的存储空间？（ ）。
A. 4MB B. 8MB C. 32MB D. 16MB
5. 冒泡排序算法的伪代码如下：
输入：数组 L , $n \geq 1$ 。输出：按非递减顺序排序的 L 。
算法 BubbleSort:
1. $FLAG \leftarrow n$ //标记被交换的最后元素位置
2. $while FLAG > 1$ do
3. $k \leftarrow FLAG - 1$
4. $FLAG \leftarrow 1$
5. for $j=1$ to k do
6. if $L(j) > L(j+1)$ then do
7. $L(j) \leftrightarrow L(j+1)$
8. $FLAG \leftarrow j$



对 n 个数用以上冒泡排序算法进行排序，最少需要比较多少次？（ ）。

- A. n B. $n-2$ C. n^2 D. $n-1$

6. 设 A 是 n 个实数的数组，考虑下面的递归算法：

```
XYZ (A[1..n])
1. if n=1 then return A[1]
2. else temp ← XYZ (A[1..n-1])
3.     if temp < A[n]
4.         then return temp
5.     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 数组的最大值 | D. A 数组的中值 |

7. 链表不具有的特点是（ ）。

- | | |
|----------------|------------------|
| A. 插入删除不需要移动元素 | B. 可随机访问任一元素 |
| C. 不必事先估计存储空间 | D. 所需空间与线性表长度成正比 |

8. 有 10 个顶点的无向图至少应该有（ ）条边才能确保是一个连通图。

- A. 10 B. 12 C. 9 D. 11

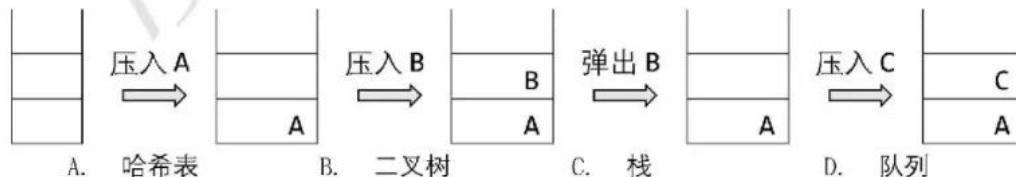
9. 二进制数 1011 转换成十进制数是（ ）。

- A. 10 B. 13 C. 11 D. 12

10. 五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. 24 B. 36 C. 72 D. 48

11. 下图中所使用的数据结构是（ ）。



- A. 哈希表 B. 二叉树 C. 栈 D. 队列

12. 独根树的高度为 1。具有 61 个结点的完全二叉树的高度为（ ）。

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

13. 干支纪年法是中国传统的纪年方法，由 10 个天干和 12 个地支组合成 60 个天干地支。由公历年份可以根据以下公式和表格换算出对应的天干地支。

天干 = (公历年份) 除以 10 所得余数

地支 = (公历年份) 除以 12 所得余数



| | | | | | | | | | | | | |
|----|---|---|---|---|---|---|----|----|---|---|---|---|
| 天干 | 甲 | 乙 | 丙 | 丁 | 戊 | 己 | 庚 | 辛 | 壬 | 癸 | | |
| | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | | |
| 地支 | 子 | 丑 | 寅 | 卯 | 辰 | 巳 | 午 | 未 | 申 | 酉 | 戌 | 亥 |
| | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 0 | 1 | 2 | 3 |

例如，今年是 2020 年， $2020 \div 10$ 余数为 0，查表为“庚”； $2020 \div 12$ ，余数为 4，查表为“子”，所以今年是庚子年。

请问 1949 年的天干地支是（ ）

- A. 己亥 B. 己丑 C. 己卯 D. 己酉

14. 10 个三好学生名额分配到 7 个班级，每个班级至少有一个名额，一共有（ ）种不同的分配方案。

- A. 56 B. 84 C. 72 D. 504

15. 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. 30 B. 150 C. 180 D. 120

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 <cstdlib>
02 #include <iostream>
03 using namespace std;
04
05 char encoder[26] = {'C', 'S', 'P', 0};
06 char decoder[26];
07
08 string st;
09
10 int main() {
11     int k = 0;
12     for (int i = 0; i < 26; ++i)
13         if (encoder[i] != 0) ++k;
14     for (char x = 'A'; x <= 'Z'; ++x) {
15         bool flag = true;
16         for (int i = 0; i < 26; ++i)
17             if (encoder[i] == x) {
18                 flag = false;
19                 break;

```



```

20      }
21      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

- 1) 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。
()
- 2) 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。
()
- 3) 将第 12 行的“`i < 26`”改为“`i < 16`”，程序运行结果不会改变。
()
- 4) 将第 26 行的“`i < 26`”改为“`i < 16`”，程序运行结果不会改变。
()

● 单选题

- 5) 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 A 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 S 又有 P
 - D. 输入的字符串中既有 A 又有 B
- 6) 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 J 又有 R
 - B. 输入的字符串中既有 P 又有 K
 - C. 输入的字符串中既有 J 又有 K
 - D. 输入的字符串中既有 P 又有 R

2.

01 #include <iostream>



```

02 using namespace std;
03
04 long long n, ans;
05 int k, len;
06 long long d[1000000];
07
08 int main() {
09     cin >> n >> k;
10     d[0] = 0;
11     len = 1;
12     ans = 0;
13     for (long long i = 0; i < n; ++i) {
14         ++d[0];
15         for (int j = 0; j + 1 < len; ++j) {
16             if (d[j] == k) {
17                 d[j] = 0;
18                 d[j + 1] += 1;
19                 ++ans;
20             }
21         }
22         if (d[len - 1] == k) {
23             d[len - 1] = 0;
24             d[len] = 1;
25             ++len;
26             ++ans;
27         }
28     }
29     cout << ans << endl;
30     return 0;
31 }
```

假设输入的 n 是不超过 2^{62} 的正整数， k 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

- 1) 若 $k=1$ ，则输出 ans 时， $len=n$ 。 ()
- 2) 若 $k>1$ ，则输出 ans 时， len 一定小于 n 。 ()
- 3) 若 $k>1$ ，则输出 ans 时， k^{len} 一定大于 n 。 ()

● 单选题

- 4) 若输入的 n 等于 10^{15} ，输入的 k 为 1，则输出等于 ()。

| | | | |
|--------------------------|--------------------------|------|--------------|
| A. $(10^{30}-10^{15})/2$ | B. $(10^{30}+10^{15})/2$ | C. 1 | D. 10^{15} |
|--------------------------|--------------------------|------|--------------|



- 5) 若输入的 n 等于 $205,891,132,094,649$ （即 3^{30} ），输入的 k 为 3，则输出等于（ ）。
- A. $(3^{30}-1)/2$ B. 3^{30} C. $3^{30}-1$ D. $(3^{30}+1)/2$
- 6) 若输入的 n 等于 $100,010,002,000,090$ ，输入的 k 为 10，则输出等于（ ）。
- A. $11,112,222,444,543$ B. $11,122,222,444,453$
 C. $11,122,222,444,543$ D. $11,112,222,444,453$

3.

```

01 #include <algorithm>
02 #include <iostream>
03 using namespace std;
04
05 int n;
06 int d[50][2];
07 int ans;
08
09 void dfs(int n, int sum) {
10    if (n == 1) {
11        ans = max(sum, ans);
12        return;
13    }
14    for (int i = 1; i < n; ++i) {
15        int a = d[i - 1][0], b = d[i - 1][1];
16        int x = d[i][0], y = d[i][1];
17        d[i - 1][0] = a + x;
18        d[i - 1][1] = b + y;
19        for (int j = i; j < n - 1; ++j)
20            d[j][0] = d[j + 1][0], d[j][1] = d[j + 1][1];
21        int s = a + x + abs(b - y);
22        dfs(n - 1, sum + s);
23        for (int j = n - 1; j > i; --j)
24            d[j][0] = d[j - 1][0], d[j][1] = d[j - 1][1];
25        d[i - 1][0] = a, d[i - 1][1] = b;
26        d[i][0] = x, d[i][1] = y;
27    }
28 }
29
30 int main() {
31    cin >> n;
32    for (int i = 0; i < n; ++i)

```



```

33     cin >> d[i][0];
34     for (int i = 0; i < n; ++i)
35         cin >> d[i][1];
36     ans = 0;
37     dfs(n, 0);
38     cout << ans << endl;
39     return 0;
40 }

```

假设输入的 n 是不超过 50 的正整数， $d[i][0]$ 、 $d[i][1]$ 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

- 1) 若输入 n 为 0，此程序可能会死循环或发生运行错误。 ()
- 2) 若输入 n 为 20，接下来的输入全为 0，则输出为 0。 ()
- 3) 输出的数一定不小于输入的 $d[i][0]$ 和 $d[i][1]$ 的任意一个。 ()

● 单选题

- 4) 若输入的 n 为 20，接下来的输入是 20 个 9 和 20 个 0，则输出为 ()。

| | | | |
|---------|---------|---------|---------|
| A. 1917 | B. 1908 | C. 1881 | D. 1890 |
|---------|---------|---------|---------|
- 5) 若输入的 n 为 30，接下来的输入是 30 个 0 和 30 个 5，则输出为 ()。

| | | | |
|---------|---------|---------|---------|
| A. 2020 | B. 2030 | C. 2010 | D. 2000 |
|---------|---------|---------|---------|
- 6) (4 分) 若输入的 n 为 15，接下来的输入是 15 到 1，以及 15 到 1，则输出为 ()。

| | | | |
|---------|---------|---------|---------|
| A. 2420 | B. 2220 | C. 2440 | D. 2240 |
|---------|---------|---------|---------|

三、完善程序（单选题，每小题 3 分，共计 30 分）

1. (质因数分解) 给出正整数 n ，请输出将 n 质因数分解的结果，结果从小到大输出。

例如：输入 $n=120$ ，程序应该输出 2 2 2 3 5，表示 $120=2\times2\times2\times3\times5$ 。输入保证 $2 \leq n \leq 10^9$ 。提示：先从小到大枚举变量 i ，然后用 i 不停试除 n 来寻找所有的质因数了。

试补全程序。

```

01 #include <cstdio>
02 using namespace std;

```



```

03
04 int n, i;
05
06 int main() {
07     scanf("%d", &n);
08     for(i = ①; ② <= n; i++) {
09         ③ {
10             printf("%d ", i);
11             n = n / i;
12         }
13     }
14     if(④)
15         printf("%d ", ⑤);
16     return 0;
17 }

```

- 1) ①处应填 ()
A. $n - 1$ B. 0 C. 1 D. 2
- 2) ②处应填 ()
A. n / i B. $n / (i * i)$ C. $i * i * i$ D. $i * i$
- 3) ③处应填 ()
A. `if (i * i <= n)` B. `if (n % i == 0)`
C. `while (i * i <= n)` D. `while (n % i == 0)`
- 4) ④处应填 ()
A. $n > 1$ B. $n <= 1$ C. $i + i <= n$ D. $i < n / i$
- 5) ⑤处应填 ()
A. 2 B. i C. n / i D. n

2. (最小区间覆盖)给出 n 个区间，第 i 个区间的左右端点是 $[a_i, b_i]$ 。现在要在这些区间中选出若干个，使得区间 $[0, m]$ 被所选区间的并覆盖（即每一个 $0 \leq i \leq m$ 都在某个所选的区间中）。保证答案存在，求所选区间个数的最小值。

输入第一行包含两个整数 n 和 m ($1 \leq n \leq 5000, 1 \leq m \leq 10^9$)。
接下来 n 行，每行两个整数 a_i, b_i ($0 \leq a_i, b_i \leq m$)。

提示：使用贪心法解决这个问题。先用 $\Theta(n^2)$ 的时间复杂度排序，然后贪心选择这些区间。

试补全程序。



```

01 #include <iostream>
02
03 using namespace std;
04
05 const int MAXN = 5000;
06 int n, m;
07 struct segment { int a, b; } A[MAXN];
08
09 void sort() // 排序
10 {
11     for (int i = 0; i < n; i++)
12         for (int j = 1; j < n; j++)
13             if (①)
14             {
15                 segment t = A[j];
16                 ②
17             }
18 }
19
20 int main()
21 {
22     cin >> n >> m;
23     for (int i = 0; i < n; i++)
24         cin >> A[i].a >> A[i].b;
25     sort();
26     int p = 1;
27     for (int i = 1; i < n; i++)
28         if (③)
29             A[p++] = A[i];
30     n = p;
31     int ans = 0, r = 0;
32     int q = 0;
33     while (r < m)
34     {
35         while (④)
36             q++;
37         ⑤;
38         ans++;
39     }
40     cout << ans << endl;
41     return 0;
42 }
```



1) ①处应填 ()

- A. $A[j].b < A[j - 1].b$ B. $A[j].b > A[j - 1].b$
 C. $A[j].a < A[j - 1].a$ D. $A[j].a > A[j - 1].a$

2) ②处应填 ()

- A. $A[j - 1] = A[j]; A[j] = t;$
 B. $A[j + 1] = A[j]; A[j] = t;$
 C. $A[j] = A[j - 1]; A[j - 1] = t;$
 D. $A[j] = A[j + 1]; A[j + 1] = t;$

3) ③处应填 ()

- A. $A[i].b < A[p - 1].b$ B. $A[i].b > A[i - 1].b$
 C. $A[i].b > A[p - 1].b$ D. $A[i].b < A[i - 1].b$

4) ④处应填 ()

- A. $q + 1 < n \&& A[q + 1].b \leq r$
 B. $q + 1 < n \&& A[q + 1].a \leq r$
 C. $q < n \&& A[q].a \leq r$
 D. $q < n \&& A[q].b \leq r$

5) ⑤处应填 ()

- A. $r = \max(r, A[q + 1].a)$ B. $r = \max(r, A[q].b)$
 C. $r = \max(r, A[q + 1].b)$ D. $q++$

CSP-J1 参考答案

单项选择

AADCC BAAAA ADCAA

阅读程序

- 1) ✓ ✗ ✓ ✗ A D
- 2) ✗ ✗ ✓ D B D
- 3) ✗ ✓ ✗ B C C

完善程序

CCCAC BDAAB

CSP-S1 参考答案

单项选择

CBBBD BAACC CDBDC

阅读程序

1. ✗ ✗ ✓ ✓ C C
2. ✗ ✓ 都对 B A D
3. ✓ ✗ ✗ D D C

完善程序

DBDDB DBCAB

CSP-J 2020 入门级第二轮试题

2020 年 CCF 非专业级软件能力认证入门组第二轮

优秀的拆分（power）

【题目描述】

一般来说，一个正整数可以拆分成若干个正整数的和。例如， $1 = 1$ ， $10 = 1 + 2 + 3 + 4$ 等。

对于正整数 n 的一种特定拆分，我们称它为“优秀的”，当且仅当在这种拆分下， n 被分解为了若干个不同的 2 的正整数次幂。注意，一个数 x 能被表示成 2 的正整数次幂，当且仅当 x 能通过正整数个 2 相乘在一起得到。

例如， $10 = 8 + 2 = 2^3 + 2^1$ 是一个优秀的拆分。但是， $7 = 4 + 2 + 1 = 2^2 + 2^1 + 2^0$ 就不是一个优秀的拆分，因为 1 不是 2 的正整数次幂。

现在，给定正整数 n ，你需要判断这个数的所有拆分中，是否存在优秀的拆分。若存在，请你给出具体的拆分方案。

【输入格式】

输入文件名为 `power.in`。

输入文件只有一行，一个正整数 n ，代表需要判断的数。

【输出格式】

输出文件名为 `power.out`。

如果这个数的所有拆分中，存在优秀的拆分。那么，你需要从大到小输出这个拆分中的每一个数，相邻两个数之间用一个空格隔开。可以证明，在规定了拆分数字的顺序后，该拆分方案是唯一的。

若不存在优秀的拆分，输出“-1”（不包含双引号）。

【样例 1 输入】

6

【样例 1 输出】

4 2

【样例 1 解释】

$6 = 4 + 2 = 2^2 + 2^1$ 是一个优秀的拆分。注意， $6 = 2 + 2 + 2$ 不是一个优秀的拆分，因为拆分成的 3 个数不满足每个数互不相同。

【样例 2 输入】

7

2020 年 CCF 非专业级软件能力认证入门组第二轮

【样例 2 输出】

-1

【样例 3】

见选手目录下的 power/power3.in 与 power/power3.ans。

【数据范围与提示】

对于 20% 的数据， $n \leq 10$ 。

对于另外 20% 的数据，保证 n 为奇数。

对于另外 20% 的数据，保证 n 为 2 的正整数次幂。

对于 80% 的数据， $n \leq 1024$ 。

对于 100% 的数据， $1 \leq n \leq 1 \times 10^7$ 。



2020 年 CCF 非专业级软件能力认证入门组第二轮

直播获奖（live）

【题目描述】

NOI2130 即将举行。为了增加观赏性，CCF 决定逐一评出每个选手的成绩，并直播即时的获奖分数线。本次竞赛的获奖率为 $w\%$ ，即当前排名前 $w\%$ 的选手的最低成绩就是即时的分数线。

更具体地，若当前已评出了 p 个选手的成绩，则当前计划获奖人数为 $\max(1, \lfloor p \times w\% \rfloor)$ ，其中 w 是获奖百分比， $\lfloor x \rfloor$ 表示对 x 向下取整， $\max(x, y)$ 表示 x 和 y 中较大的数。如有选手成绩相同，则所有成绩并列的选手都能获奖，因此实际获奖人数可能比计划中多。

作为评测组的技术人员，请你帮 CCF 写一个直播程序。

【输入格式】

输入文件名为 live.in。

第 1 行两个正整数 n, w 。分别代表选手总数与获奖率。

第 2 行有 n 个非负整数，依次代表逐一评出的选手成绩。

【输出格式】

输出文件名为 live.out。

只有一行，包含 n 个非负整数，依次代表选手成绩逐一评出后，即时的获奖分数线。相邻两个整数间用一个空格分隔。

【样例 1 输入】

```
10 60
200 300 400 500 600 600 0 300 200 100
```

【样例 1 输出】

```
200 300 400 400 400 500 400 400 300 300
```

【样例 1 解释】

| | | | | | | | | | | |
|---------|---|---|---|---|---|---|---|---|---|----|
| 已评测选手人数 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 计划获奖人数 | 1 | 1 | 1 | 2 | 3 | 3 | 4 | 4 | 5 | 6 |

2020 年 CCF 非专业级软件能力认证入门组第二轮

| | | | | | | | | | | |
|--|------------|------------|------------|------------|------------|------------|------------|------------|------------|-----|
| 已评测选手的分 数从高到低排列 (其中，分数线 用 粗体 标出) | 200 | 300 | 400 | 500 | 600 | 600 | 600 | 600 | 600 | 600 |
| | 200 | 300 | 400 | 500 | 600 | 600 | 600 | 600 | 600 | 600 |
| | 200 | 300 | 400 | 500 | 500 | 500 | 500 | 500 | 500 | 500 |
| | 200 | 300 | 400 | 400 | 400 | 400 | 400 | 400 | 400 | 400 |
| | 200 | 300 | 300 | 200 | 300 | 300 | 300 | 300 | 300 | 300 |
| | 200 | 200 | 200 | 200 | 200 | 200 | 300 | 300 | 300 | 300 |
| | 0 | | | | 0 | 200 | 200 | 200 | 200 | 200 |
| | | | | | 0 | 200 | 200 | 200 | 200 | 100 |
| | | | | | | 0 | | | | 0 |

注意，在第 9 名选手的成绩评出之后，计划获奖人数为 5 人，但由于有并列，因此实际会有 6 人获奖。

【样例 2 输入】

10 30
100 100 600 100 100 100 100 100 100 100

【样例 2 输出】

100 100 600 600 600 100 100 100 100

【样例 3】

见选手目录下的 live/live3.in 与 live/live3.ans。

【数据范围与提示】

| 测试点编号 | n |
|-------|----------|
| 1~3 | = 10 |
| 4~6 | = 500 |
| 7~10 | = 2000 |
| 11~17 | = 10000 |
| 18~20 | = 100000 |

对于所有测试点，每个选手的成绩均为不超过 600 的非负整数，获奖百分比 w 是一个正整数且 $1 \leq w \leq 99$ 。

在计算计划获奖人数时，如用浮点类型的变量（如 C/C++ 中的 float、double，Pascal 中的 real、double、extended 等）存储获奖比例 $w\%$ ，则计算 $5 \times 60\%$ 时的结果可能为 3.000001，也可能为 2.999999，向下取整后的结果不确定。因此，建议仅使用整型变量，以计算出准确值。

2020 年 CCF 非专业级软件能力认证入门组第二轮

表达式 (expr)

【题目描述】

小 C 热衷于学习数理逻辑。有一天，他发现了一种特别的逻辑表达式。在这种逻辑表达式中，所有操作数都是变量，且它们的取值只能为 0 或 1，运算从左往右进行。如果表达式中有括号，则先计算括号内的子表达式的值。特别的，这种表达式有且仅有以下几种运算：

1. 与运算： $a \& b$ 。当且仅当 a 和 b 的值都为 1 时，该表达式的值为 1。其余情况该表达式的值为 0。
2. 或运算： $a | b$ 。当且仅当 a 和 b 的值都为 0 时，该表达式的值为 0。其余情况该表达式的值为 1。
3. 取反运算： $!a$ 。当且仅当 a 的值为 0 时，该表达式的值为 1。其余情况该表达式的值为 0。

小 C 想知道，给定一个逻辑表达式和其中每一个操作数的初始取值后，再取反某一个操作数的值时，原表达式的值为多少。

为了化简对表达式的处理，我们有如下约定：

表达式将采用后缀表达式的方式输入。后缀表达式的定义如下：

1. 如果 E 是一个操作数，则 E 的后缀表达式是它本身。
2. 如果 E 是 $E_1 op E_2$ 形式的表达式，其中 op 是任何二元操作符，且优先级不高于 E_1 、 E_2 中括号外的操作符，则 E 的后缀式为 $E'_1 E'_2 op$ ，其中 E'_1 、 E'_2 分别为 E_1 、 E_2 的后缀式。
3. 如果 E 是 (E_1) 形式的表达式，则 E_1 的后缀式就是 E 的后缀式。

同时为了方便，输入中：

- a) 与运算符 ($\&$)、或运算符 ($|$)、取反运算符 ($!$) 的左右均有一个空格，但表达式末尾没有空格。
- b) 操作数由小写字母 x 与一个正整数拼接而成，正整数表示这个变量的下标。例如： x_{10} ，表示下标为 10 的变量 x_{10} 。数据保证每个变量在表达式中出现恰好一次。

【输入格式】

输入文件名为 `expr.in`。

第一行包含一个字符串 s ，表示上文描述的表达式。

第二行包含一个正整数 n ，表示表达式中变量的数量。表达式中变量的下标为 $1, 2, \dots, n$ 。

第三行包含 n 个整数，第 i 个整数表示变量 x_i 的初值。

第四行包含一个正整数 q ，表示询问的个数。

接下来 q 行，每行一个正整数，表示需要取反的变量的下标。注意，每一个询问的修改都是临时的，即之前询问中的修改不会对后续的询问造成影响。

数据保证输入的表达式合法。变量的初值为 0 或 1。

2020 年 CCF 非专业级软件能力认证入门组第二轮

【输出格式】

输出文件名为 `expr.out`。

输出一共有 q 行，每行一个 0 或 1，表示该询问下表达式的值。

【样例 1 输入】

```
x1 x2 & x3 |
3
1 0 1
3
1
2
3
```

【样例 1 输出】

```
1
1
0
```

【样例 1 解释】

该后缀表达式的中缀表达式形式为 $(x_1 \& x_2) | x_3$ 。

对于第一次询问，将 x_1 的值取反。此时，三个操作数对应的赋值依次为 0, 0, 1。原表达式的值为 $(0 \& 0) | 1 = 1$ 。

对于第二次询问，将 x_2 的值取反。此时，三个操作数对应的赋值依次为 1, 1, 1。原表达式的值为 $(1 \& 1) | 1 = 1$ 。

对于第三次询问，将 x_3 的值取反。此时，三个操作数对应的赋值依次为 1, 0, 0。原表达式的值为 $(1 \& 0) | 0 = 0$ 。

【样例 2 输入】

```
x1 ! x2 x4 | x3 x5 ! & & ! &
5
0 1 0 1 1
3
1
3
5
```

【样例 2 输出】

2020 年 CCF 非专业级软件能力认证入门组第二轮

0
1
1

【样例 2 解释】

该表达式的中缀表达式形式为 $(!x_1) \& (!((x_2 | x_4) \& (x_3 \& (!x_5))))$ 。

【样例 3】

见选手目录下的 `expr/expr3.in` 与 `expr/expr3.ans`。

【数据范围与提示】

对于 20% 的数据，表达式中有且仅有与运算 ($\&$) 或者或运算 ($|$)。

对于另外 30% 的数据， $|s| \leq 1000$, $q \leq 1000$, $n \leq 1000$ 。

对于另外 20% 的数据，变量的初值全为 0 或全为 1。

对于 100% 的数据， $1 \leq |s| \leq 1 \times 10^6$, $1 \leq q \leq 1 \times 10^5$, $2 \leq n \leq 1 \times 10^5$ 。

其中， $|s|$ 表示字符串 s 的长度。



2020 年 CCF 非专业级软件能力认证入门组第二轮

方格取数（number）

【题目描述】

设有 $n \times m$ 的方格图，每个方格中都有一个整数。现有一只小熊，想从图的左上角走到右下角，每一步只能向上、向下或向右走一格，并且不能重复经过已经走过的方格，也不能走出边界。小熊会取走所有经过的方格中的整数，求它能取到的整数之和的最大值。

【输入格式】

输入文件名为 number.in。

第 1 行两个正整数 n, m 。

接下来 n 行每行 m 个整数，依次代表每个方格中的整数。

【输出格式】

输入文件名为 number.out。

一个整数，表示小熊能取到的整数之和的最大值。

【样例 1 输入】

```
3 4
1 -1 3 2
2 -1 4 -1
-2 2 -3 -1
```

【样例 1 输出】

9

【样例 1 解释】

| | | | |
|----|------|-----|-----|
| 1 | -1 | 3 | → 2 |
| 2 | → -1 | → 4 | -1 |
| -2 | 2 | -3 | -1 |

按上述走法，取到的数之和为 $1 + 2 + (-1) + 4 + 3 + 2 + (-1) + (-1) = 9$ ，可以证明为最大值。

2020 年 CCF 非专业级软件能力认证入门组第二轮

| | | | |
|----|------------|-----|------|
| 1 | -1 | 3 | → 2 |
| ↓ | 2 → -1 → 4 | 3 ↑ | ↓ -1 |
| -2 | 2 | -3 | ↓ -1 |

注意，上述走法是错误的，因为第 2 行第 2 列的方格走过了两次，而根据题意，不能重复经过已经走过的方格。

| | | | |
|----|------------|-----|------|
| 1 | -1 | 3 | → 2 |
| ↓ | 2 → -1 → 4 | 3 ↑ | ↓ -1 |
| -2 | 2 | -3 | ↓ -1 |

另外，上述走法也是错误的，因为没有走到右下角的终点。

【样例 2 输入】

2 5
-1 -1 -3 -2 -7
-2 -1 -4 -1 -2

【样例 2 输出】

-10

【样例 2 解释】

| | | |
|-------------------|----|----|
| -1 → -1 → -3 → -2 | ↓ | -7 |
| -2 | -1 | -4 |

按上述走法，取到的数之和为 $(-1) + (-1) + (-3) + (-2) + (-1) + (-2) = -10$ ，可以证明为最大值。因此，请注意，取到的数之和的最大值也可能是负数。

【样例 3】

见选手目录下的 number/number3.in 与 number/number3.ans。

【数据范围与提示】

对于 20% 的数据， $n, m \leq 5$ 。

对于 40% 的数据， $n, m \leq 50$ 。

对于 70% 的数据， $n, m \leq 300$ 。

对于 100% 的数据， $1 \leq n, m \leq 1000$ 。方格中整数的绝对值不超过 10^4 。

CSP-S 2020 提高级第二轮试题

2020 年 CCF 非专业级软件能力认证 提高级第二轮

2020 CCF CSP-S2

时间：2020 年 11 月 7 日 14:30 ~ 18:30

| | | | | |
|--------|------------|---------|----------|------------|
| 题目名称 | 儒略日 | 动物园 | 函数调用 | 贪吃蛇 |
| 题目类型 | 传统型 | 传统型 | 传统型 | 传统型 |
| 目录 | julian | zoo | call | snakes |
| 可执行文件名 | julian | zoo | call | snakes |
| 输入文件名 | julian.in | zoo.in | call.in | snakes.in |
| 输出文件名 | julian.out | zoo.out | call.out | snakes.out |
| 时间限制 | 1.0 秒 | 1.0 秒 | 2.0 秒 | 2.0 秒 |
| 内存限制 | 256 MB | 256 MB | 256 MB | 256 MB |
| 测试点数目 | 10 | 20 | 20 | 20 |

提交源程序文件名

| | | | | |
|-----------|------------|---------|----------|------------|
| C++ 语言 | julian.cpp | zoo.cpp | call.cpp | snakes.cpp |
| C 语言 | julian.c | zoo.c | call.c | snakes.c |
| Pascal 语言 | julian.pas | zoo.pas | call.pas | snakes.pas |

编译选项

| | |
|-----------|-----|
| C++ 语言 | -lm |
| C 语言 | -lm |
| Pascal 语言 | |

注意事项（请选手仔细阅读）

1. 文件名（程序名和输入输出文件名）必须使用英文小写。
2. C/C++ 中函数 `main()` 的返回值类型必须是 `int`，程序正常结束时的返回值必须是 0。
3. 提交的程序代码文件的放置位置请参照各省的具体要求。
4. 因违反以上三点而出现的错误或问题，申诉时一律不予受理。
5. 若无特殊说明，结果的比较方式为全文比较（过滤行末空格及文末回车）。
6. 程序可使用的栈内存空间限制与题目的内存限制一致。
7. 全国统一评测时采用的机器配置为：Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz，内存 32GB。上述时限以此配置为准。
8. 只提供 Linux 格式附加样例文件。
9. 评测在当前最新公布的 NOI Linux 下进行，各语言的编译器版本以其为准。

儒略日 (julian)

【题目描述】

为了简便计算，天文学家们使用儒略日（Julian day）来表达时间。所谓儒略日，其定义为从公元前 4713 年 1 月 1 日正午 12 点到此后某一时刻间所经过的天数，不满一天者用小数表达。若利用这一天文历法，则每一个时刻都将被均匀的映射到数轴上，从而得以很方便的计算它们的差值。

现在，给定一个不含小数部分的儒略日，请你帮忙计算出该儒略日（一定是某一天的中午 12 点）所对应的公历日期。

我们现行的公历为格里高利历（Gregorian calendar），它是在公元 1582 年由教皇格里高利十三世在原有的儒略历（Julian calendar）的基础上修改得到的（注：儒略历与儒略日并无直接关系）。具体而言，现行的公历日期按照以下规则计算：

1. 公元 1582 年 10 月 15 日（含）以后：适用格里高利历，每年一月 31 天、二月 28 天或 29 天、三月 31 天、四月 30 天、五月 31 天、六月 30 天、七月 31 天、八月 31 天、九月 30 天、十月 31 天、十一月 30 天、十二月 31 天。其中，闰年的二月为 29 天，平年为 28 天。当年份是 400 的倍数，或日期年份是 4 的倍数但不是 100 的倍数时，该年为闰年。
2. 公元 1582 年 10 月 5 日（含）至 10 月 14 日（含）：不存在，这些日期被删除，该年 10 月 4 日之后为 10 月 15 日。
3. 公元 1582 年 10 月 4 日（含）以前：适用儒略历，每月天数与格里高利历相同，但只要年份是 4 的倍数就是闰年。
4. 尽管儒略历于公元前 45 年才开始实行，且初期经过若干次调整，但今天人类习惯于按照儒略历最终的规则反推一切 1582 年 10 月 4 日之前的时间。
注意，公元零年并不存在，即公元前 1 年的下一年是公元 1 年。因此公元前 1 年、前 5 年、前 9 年、前 13 年……以此类推的年份应视为闰年。

【输入格式】

输入文件名为 julian.in。

第一行一个整数 Q ，表示询问的组数。

接下来 Q 行，每行一个非负整数 r_i ，表示一个儒略日。

【输出格式】

输出文件名为 julian.out。

对于每一个儒略日 r_i ，输出一行表示日期的字符串 s_i 。共计 Q 行。

s_i 的格式如下：

1. 若年份为公元后，输出格式为 Day Month Year。其中日（Day）、月（Month）、年（Year）均不含前导零，中间用一个空格隔开。例如：公元 2020 年 11 月 7 日正午 12 点，输出为 7 11 2020。

2020 年 CCF 非专业级软件能力认证提高级第二轮

2. 若年份为公元前，输出格式为 Day Month Year BC。其中年（Year）输出该年份的数值，其余与公元后相同。例如：公元前 841 年 2 月 1 日正午 12 点，输出为 12 841 BC。

【样例 1 输入】

```
3
10
100
1000
```

【样例 1 输出】

```
11 1 4713 BC
10 4 4713 BC
27 9 4711 BC
```

【样例 2 输入】

```
3
2000000
3000000
4000000
```

【样例 2 输出】

```
14 9 763
15 8 3501
12 7 6239
```

【样例 3】

见选手目录下的 julian/julian3.in 与 julian/julian3.ans。

【数据范围与提示】

| 测试点编号 | $Q =$ | $r_i \leq$ |
|-------|-------|------------|
| 1 | 1000 | 365 |
| 2 | | 10^4 |
| 3 | | 10^5 |

2020 年 CCF 非专业级软件能力认证提高级第二轮

| | | |
|----|---------------------|-------------------|
| 4 | 10000 10^5 | 3×10^5 |
| 5 | | 2.5×10^6 |
| 6 | | 5×10^6 |
| 7 | | 10^7 |
| 8 | | 10^9 |
| 9 | | |
| 10 | | 答案年份不超过 10^9 |



2020 年 CCF 非专业级软件能力认证提高级第二轮

动物园 (zoo)

【题目描述】

动物园里饲养了很多动物，饲养员小 A 会根据饲养动物的情况，按照《饲养指南》购买不同种类的饲料，并将购买清单发给采购员小 B。

具体而言，动物世界里存在 2^k 种不同的动物，它们被编号为 $0 \sim 2^k - 1$ 。动物园里饲养了其中的 n 种，其中第 i 种动物的编号为 a_i 。

《饲养指南》中共有 m 条要求，第 j 条要求形如“如果动物园中饲养着某种动物，满足其编号的二进制表示的第 p_j 位为 1，则必须购买第 q_j 种饲料”。其中饲料共有 c 种，它们从 $1 \sim c$ 编号。本题中我们将动物编号的二进制表示视为一个 k 位 01 串，第 0 位是最低位，第 $k - 1$ 位是最高位。

根据《饲养指南》，小 A 将会制定饲料清单交给小 B，由小 B 购买饲料。清单形如一个 c 位 01 串，第 i 位为 1 时，表示需要购买第 i 种饲料；第 i 位为 0 时，表示不需要购买第 i 种饲料。

实际上根据购买到的饲料，动物园可能可以饲养更多的动物。更具体地，如果将当前未被饲养的编号为 x 的动物加入动物园饲养后，饲料清单没有变化，那么我们认为动物园当前还能饲养编号为 x 的动物。

现在小 B 想请你帮忙算算，动物园目前还能饲养多少种动物。

【输入格式】

输入文件名为 `zoo.in`。

第一行包含四个以空格分隔的整数 n 、 m 、 c 、 k 。分别表示动物园中动物数量、《饲养指南》要求数、饲料种数与动物编号的二进制表示位数。

第二行 n 个以空格分隔的整数，其中第 i 个整数表示 a_i 。

接下来 m 行，每行两个整数 p_i 、 q_i 表示一条要求。

数据保证所有 a_i 互不相同，所有的 q_i 互不相同。

【输出格式】

输出文件名为 `zoo.out`。

仅一行一个整数表示答案。

【样例 1 输入】

```
3 3 5 4
1 4 6
0 3
2 4
2 5
```

2020 年 CCF 非专业级软件能力认证提高级第二轮

【样例 1 输出】

13

【样例 1 解释】

动物园里饲养了编号为 1、4、6 的三种动物，《饲养指南》上 3 条要求为：

1. 若饲养的某种动物的编号的第 0 个二进制位为 1，则需购买第 3 种饲料。
2. 若饲养的某种动物的编号的第 2 个二进制位为 1，则需购买第 4 种饲料。
3. 若饲养的某种动物的编号的第 2 个二进制位为 1，则需购买第 5 种饲料。

饲料购买情况为：

1. 编号为 1 的动物的第 0 个二进制位为 1，因此需要购买第 3 种饲料；
2. 编号为 4、6 的动物的第 2 个二进制位为 1，因此需要购买第 4、5 种饲料。

由于在当前动物园中加入一种编号为 $0, 2, 3, 5, 7, 8, \dots, 15$ 之一的动物，购物清单都不会改变，因此答案为 13。

【样例 2 输入】

2 2 4 3
1 2
1 3
2 4

【样例 2 输出】

2

【样例 3】

见选手目录下的 `zoo/zoo3.in` 与 `zoo/zoo3.ans`。

【数据范围与提示】

对于 20% 的数据： $k \leq n \leq 5$, $m \leq 10$, $c \leq 10$, 所有的 p_i 互不相同。

对于 40% 的数据： $n \leq 15$, $k \leq 20$, $m \leq 20$, $c \leq 20$ 。

对于 60% 的数据： $n \leq 30$, $k \leq 30$, $m \leq 1000$ 。

对于 100% 的数据： $0 \leq n, m \leq 10^6$, $0 \leq k \leq 64$, $1 \leq c \leq 10^8$ 。

2020 年 CCF 非专业级软件能力认证提高级第二轮

函数调用（call）

【题目描述】

函数是各种编程语言中一项重要的概念，借助函数，我们总可以将复杂的任务分解成一个个相对简单的子任务，直到细化为十分简单的基础操作，从而使代码的组织更加严密、更加有条理。然而，过多的函数调用也会导致额外的开销，影响程序的运行效率。

某数据库应用程序提供了若干函数用以维护数据。已知这些函数的功能可分为三类：

1. 将数据中的指定元素加上一个值；
2. 将数据中的每一个元素乘以一个相同值；
3. 依次执行若干次函数调用，保证不会出现递归（即不会直接或间接地调用本身）。

在使用该数据库应用时，用户可一次性输入要调用的函数序列（一个函数可能被调用多次），在依次执行完序列中的函数后，系统中的数据被加以更新。某一天，小 A 在应用该数据库程序处理数据时遇到了困难：由于频繁而低效的函数调用，系统在执行操作时进入了无响应的状态，他只好强制结束了数据库程序。为了计算出正确数据，小 A 查阅了软件的文档，了解到每个函数的具体功能信息，现在他想请你根据这些信息帮他计算出更新后的数据应该是多少。

【输入格式】

输入文件名为 `call.in`。

第 1 行一个正整数 n ，表示数据的个数。

第 2 行 n 个整数，第 i 个整数表示下标为 i 的数据的初始值为 a_i 。

第 3 行一个正整数 m ，表示数据库应用程序提供的函数个数。函数从 1 ~ m 编号。

接下来 m 行中，第 j ($1 \leq j \leq m$) 行的第一个整数为 T_j ，表示 j 号函数的类型：

1. 若 $T_j = 1$ ，接下来两个整数 P_j, V_j 分别表示要执行加法的元素的下标及其增加的值；
2. 若 $T_j = 2$ ，接下来一个整数 V_j 表示所有元素所乘的值；
3. 若 $T_j = 3$ ，接下来一个正整数 C_j 表示 j 号函数要调用的函数个数，

随后 C_j 个整数 $g_1^{(j)}, g_2^{(j)}, \dots, g_{C_j}^{(j)}$ 依次表示其所调用的函数的编号。

第 $m + 4$ 行一个正整数 Q ，表示输入的函数操作序列长度。

第 $m + 5$ 行 Q 个整数 f_i ，第 i 个整数表示第 i 个执行的函数的编号。

【输出格式】

输出文件名为 `call.out`。

2020 年 CCF 非专业级软件能力认证提高级第二轮

一行 n 个用空格隔开的整数，按照下标 $1 \sim n$ 的顺序，分别输出在执行完输入的函数序列后，数据库中每一个元素的值。答案对 998244353 取模。

【样例 1 输入】

```
3
1 2 3
3
1 1 1
2 2
3 2 1 2
2
2 3
```

【样例 1 输出】

```
6 8 12
```

【样例 1 解释】

1 号函数功能为将 a_1 的值加一。2 号函数功能为所有元素乘 2。3 号函数将先调用 1 号函数，再调用 2 号函数。

最终的函数序列先执行 2 号函数，所有元素的值变为 2,4,6。

再执行 3 号函数时，先调用 1 号函数，所有元素的值变为 3,4,6。再调用 2 号函数，所有元素的值变为 6,8,12。

【样例 2 输入】

```
10
1 2 3 4 5 6 7 8 9 10
8
3 2 2 3
3 2 4 5
3 2 5 8
2 2
3 2 6 7
1 2 5
1 7 6
2 3
3
1 2 3
```

2020 年 CCF 非专业级软件能力认证提高级第二轮

【样例 2 输出】

36 282 108 144 180 216 504 288 324 360

【样例 3】

见选手目录下的 call/call3.in 与 call/call3.ans。

【数据范围与提示】

| 测试点编号 | $n, m, Q \leq$ | $\sum C_j$ | 其他特殊限制 |
|---------|----------------|----------------------|---------------------|
| 1 ~ 2 | 1000 | = $m - 1$ | 函数调用关系构成一棵树 |
| 3 ~ 4 | | ≤ 100 | 无 |
| 5 ~ 6 | 20000 | ≤ 40000 | 不含第 2 类函数或不含第 1 类函数 |
| 7 | | = 0 | 无 |
| 8 ~ 9 | | = $m - 1$ | 函数调用关系构成一棵树 |
| 10 ~ 11 | | $\leq 2 \times 10^5$ | 无 |
| 12 ~ 13 | | $\leq 2 \times 10^5$ | 不含第 2 类函数或不含第 1 类函数 |
| 14 | | = 0 | 无 |
| 15 ~ 16 | 10^5 | = $m - 1$ | 函数调用关系构成一棵树 |
| 17 ~ 18 | | $\leq 5 \times 10^5$ | 无 |
| 19 ~ 20 | | $\leq 10^6$ | 无 |

对于所有数据： $0 \leq a_i \leq 10^4$, $T_j \in \{1, 2, 3\}$, $1 \leq P_j \leq n$, $0 \leq V_j \leq 10^4$, $1 \leq g_k^{(j)} \leq m$, $1 \leq f_i \leq m$ 。

2020 年 CCF 非专业级软件能力认证提高级第二轮

贪吃蛇（snakes）

【题目描述】

草原上有 n 条蛇，编号分别为 $1, 2, \dots, n$ 。初始时每条蛇有一个体力值 a_i ，我们称编号为 x 的蛇实力比编号为 y 的蛇强当且仅当它们当前的体力值满足 $a_x > a_y$ ，或者 $a_x = a_y$ 且 $x > y$ 。

接下来这些蛇将进行决斗，决斗将持续若干轮，每一轮实力最强的蛇拥有选择权，可以选择吃或者不吃掉实力最弱的蛇：

1. 如果选择吃，那么实力最强的蛇的体力值将减去实力最弱的蛇的体力值，实力最弱的蛇被吃掉，退出接下来的决斗。之后开始下一轮决斗。
2. 如果选择不吃，决斗立刻结束。

每条蛇希望自己不被吃的前提下在决斗中尽可能多吃别的蛇（显然，蛇不会选择吃自己）。

现在假设每条蛇都足够聪明，请你求出决斗结束后会剩几条蛇。

本题有多组数据，对于第一组数据，每条蛇体力会全部由输入给出，之后的每一组数据，会相对于上一组的数据，修改一部分蛇的体力作为新的输入。

【输入格式】

输入文件名为 `snakes.in`。

第一行一个正整数 T ，表示数据组数。

接下来有 T 组数据，对于第 1 组数据，第一行一个正整数 n ，第二行 n 个非负整数表示 a_i 。

对于第 2 组到第 T 组数据，每组数据：

第一行第一个非负整数 k 表示体力修改的蛇的个数。

第二行 $2k$ 个整数，每两个整数组成一个二元组 (x, y) ，表示依次将 a_x 的值改为 y 。一个位置可能被修改多次，以最后一次修改为准。

【输出格式】

输入文件名为 `snakes.out`。

输出 T 行，每行一个整数表示最终存活的蛇的条数。

【样例 1 输入】

```
2
3
11 14 14
3
1 5 2 6 3 25
```

2020 年 CCF 非专业级软件能力认证提高级第二轮

【样例 1 输出】

3

1

【样例 1 解释】

第一组数据，第一轮中 3 号蛇最强，1 号蛇最弱。若 3 号蛇选择吃，那么它将在第二轮被 2 号蛇吃掉。因此 3 号蛇第一轮选择不吃，3 条蛇都将存活。

对于第二组数据，3 条蛇体力变为 5,6,25。第一轮中 3 号蛇最强，1 号蛇最弱，若它选择吃，那么 3 号蛇体力值变为 20，在第二轮中依然是最强蛇并能吃掉 2 号蛇，因此 3 号蛇会选择两轮都吃，最终只有 1 条蛇存活。

【样例 2 输入】

2

5

13 31 33 39 42

5

1 7 2 10 3 24 4 48 5 50

【样例 2 输出】

5

3

【样例 3】

见选手目录下的 snakes/snakes3.in 与 snakes/snakes3.ans。

【样例 4】

见选手目录下的 snakes/snakes4.in 与 snakes/snakes4.ans。

【数据范围与提示】

对于 20% 的数据： $n = 3$ 。

对于 40% 的数据： $n \leq 10$ 。

对于 55% 的数据： $n \leq 2000$ 。

对于 70% 的数据： $n \leq 5 \times 10^4$ 。

对于 100% 的数据： $3 \leq n \leq 10^6$ ， $1 \leq T \leq 10$ ， $0 \leq k \leq 10^5$ ， $0 \leq a_i, y \leq 10^9$ 。保证每组数据（包括所有修改完成后的）的 a_i 以不降顺序排列。

NOIP 2020 在线模拟赛试题与题解

信奥题库 NOIP 2020 在线模拟赛已于 11 月 28 日顺利举行，现公开模拟赛试题、参考标程与题解，供各位选手参考。

A 题：有趣的函数

有趣的函数

输入文件名：function.in

输出文件名：function.out

共 20 个测试点，每个测试点 5 分

每个测试点限时 2 秒，运行内存上限 512MB

“阿，这漫天星光的规律，就藏在那自然对数里吧。”老 Win 感叹道。

这天他发现了一个有趣的函数 $f(x)$ ，这个函数是这样的，其中 e 是自然对数，取值约为 2.718……：

$$f(x) = f(x - e) + f(x - 2)$$

其中当 $0 \leq x < 3$ 时， $f(x) = 1$

他一眼就看穿了这个问题的简单解。

但是他想考一考你，作为即将要参加 NOIP 的高中生选手。

由于老 Win 最近很讨厌高精度算法，所以他打算直接叫你对 998244353 取模，以减轻蒟蒻出题人的烦恼。

输入格式

一个整数 T ，表示数据组数

T 行每行一个整数 n ，表示询问 $f(n)$

输出格式

T 行每行一个整数，表示 $f(n) \bmod 998244353$

数据范围

对于 10% 的数据，满足 $1 \leq n \leq 10$

对于 30% 的数据，满足 $1 \leq n \leq 1000$

对于 60% 的数据，满足 $1 \leq n \leq 100000$

对于 80% 的数据，满足 $1 \leq n \leq 1000000$

对于 100% 的数据，满足 $1 \leq n \leq 10000000, T = 5$

样例输入

```
1 5
2 10
3 1000
4 100000
5 1000000
6 10000000
```

样例输出

```
1 12
2 919123937
3 824738881
4 112429956
5 121400987
```

题解

这题十分的简朴，不过为了恶心人用了 e 。

大概就是枚举用了多少个 e ，然后剩下的用2的方案数肯定是 $O(1)$ 个的，然后就可以得出用 x 个 e 和 y 个2可以拼出来答案。

然后看一下最后一步能不能用 e 或者2来跳就可以了，答案就是个组合数的和。

参考标程

```
#include <bits/stdc++.h>
using namespace std;

const int N = 10000010;
int      T, n;
int      fac[N], inv[N];
double   e    = exp(1);
const int mod = 998244353;

int      ksm(int x, int t)
{
    int tot = 1;
    while (t) {
        if (t & 1) tot = 1ll * tot * x % mod;
        x = 1ll * x * x % mod;
        t /= 2;
    }
    return tot;
}

void ad(int& x, int y) { x = (x + y >= mod) ? (x + y - mod) : (x + y); }

int  C(int x, int y) { return 1ll * fac[x] * inv[x - y] % mod * inv[y] % mod; }

int  main()
{
    freopen("A.in", "r", stdin);
    freopen("A.out", "w", stdout);
    scanf("%d", &T);
    n      = 10000000;
    fac[0] = 1;
    for (int i = 1; i <= n; i++) fac[i] = 1ll * fac[i - 1] * i % mod;
    inv[n] = ksm(fac[n], mod - 2);
    for (int i = n - 1; i >= 0; i--) inv[i] = 1ll * inv[i + 1] * (i + 1) % mod;
}
```

```
while (T--) {
    scanf("%d", &n);
    if (n < 3) {
        printf("1\n");
        continue;
    }
    int ans = 0;
    for (int i = 0; i * e <= n; i++)
        for (int j = max((int)floor((n - i * e - 3) / 2), 0); j * 2 + i * e <= n; j++) {
            if (n - i * e - j * 2 >= 3) continue;
            if (n - i * e - (j - 1) * 2 >= 3 && j) ad(ans, C(i + j - 1, j - 1));
            if (n - (i - 1) * e - j * 2 >= 3 && i) ad(ans, C(i + j - 1, i - 1));
        }
    printf("%d\n", ans);
}
}
```

B 题：能量球

能量球

输入文件名: energy.in

输出文件名: energy.out

共 20 个测试点，每个测试点 5 分

每个测试点限时 2 秒，运行内存上限 512MB

“阿巴，我快没有电了。”老 Win 的女朋友深情的看着老 Win，呢喃着。

老 Win 的女朋友是一部智能手机，老 Win 很爱她，每时每刻都爱护着她，无论发生什么事，他都不会离开女朋友。

可是她的电量并不是普通的，可能是 $(-\infty, \infty)$ 之间的任意一个整数。

现在你手中有 n 个能量球，这 n 个能量球的能量为 $1, 2, 3, \dots, n$ 。

你可以选出若干个能量球，并给每一个选出来的能量球附上对应的能量系数，由于你没有老 Win 这么强大，所以能量系数只能为整数。

如果能量球 \times 对应的能量系数再相加恰好等于她的电量，那么老 Win 的女朋友就可以被救回来了。

由于她的电量是随机的，所以你只要满足选出的能量球存在一组能量系数使得其可以表示成任何一个整数就可以。

老 Win 手中有 N 个能量球，他当然知道如何快速救回女朋友，他只想把前 n 个能量球给你，并且问你，有多少种选出能量球的方案可以把他的女朋友救回来。

而且由于女朋友每天都会没有电，所以他想问你 T 次。

由于老 Win 最近很讨厌高精度算法，所以他打算直接叫你对 998244353 取模，以减轻蒟蒻出题人的烦恼。

输入格式

第一行一个整数 T ，表示 T 次询问。

T 行每行一个整数 n ，表示这次他将会把能量值为 1 到 n 的能量球给你。

输出格式

T 行每行一个整数表示每次方案数对 998244353 取模的结果。

数据范围

对于 20% 的数据， $n \leq 20$

对于 40% 的数据， $n \leq 1000$

对于 60% 的数据， $n \leq 30000$

对于另外 20% 的数据，满足 $T = 1$

对于 100% 的数据， $1 \leq n \leq 300000, 1 \leq T \leq 300000$

样例输入

```
1 19
2 1
3 2
4 4
5 8
6 16
7 32
8 64
9 128
10 256
11 512
12 1024
13 2048
14 4096
15 8192
16 16384
17 32768
18 65536
19 131072
20 262144
```

样例输出

```
1 1
2 2
3 11
4 236
5 65243
6 301923282
7 627961329
8 552143461
9 457396877
10 517491896
11 844068972
12 93858167
13 61592432
14 272124541
15 419448881
16 305862735
17 472197008
18 477029493
19 611084457
```

样例解释

对于 $n = 4$ 来说，除了 空集, $\{2\}$, $\{3\}$, $\{4\}$, $\{2, 4\}$ 都是可行的，所以共有 $16 - 5 = 11$ 种方案。

题解

简单的莫比乌斯反演。

我们先用 $vector$ 把每一个数的因子存下来，时间复杂度 $n \ln n$ 。

考虑构造容斥 $f(d) = 2^{t(d)} - 1$ ，其中 $t(d)$ 表示 d 的倍数有多少个， $f(d)$ 就表示有多少个集合的 gcd 的因子中有 d 。

那么答案显然就是 $\sum_{i=1}^n \mu(d) f(d)$ ，因为假若一个集合的 gcd 为 d ，那么它在因子处会被算到，系数是莫比乌斯函数，所以 $\sum_{g|d} \mu(g) = [d = 1]$ 。

将 n 离线下来维护 $f(d)$ 即可。

如果打了暴力没有加判重优化会被卡成0分，因为没有人规定 $T \leq n$ 。

参考标程

```
#include <bits/stdc++.h>
using namespace std;

const int N = 300010;
vector<int> V[N];
int mu[N], p[N], Ans[N], T, n, t[N], ci[N], ans;
bool vis[N];
const int mod = 998244353;

void read(int& x)
{
    char ch = getchar();
    x = 0;
    while (ch < '0' || ch > '9') ch = getchar();
    while (ch >= '0' && ch <= '9') x = x * 10 + ch - '0', ch = getchar();
}

void ad(int& x, int y) { x = (x + y >= mod) ? (x + y - mod) : (x + y); }

void ins(int x) { ad(ans, 1ll * mu[x] * ci[t[x]] % mod), t[x]++; }

int main()
{
    freopen("B.in", "r", stdin);
    freopen("B.out", "w", stdout);

    read(T);

    n = 300000;
    mu[1] = 1;
```

```
for (int i = 2; i <= n; i++) {
    if (!vis[i]) p[++p[0]] = i, mu[i] = mod - 1;
    for (int j = 1; j <= p[0] && 1ll * i * p[j] <= n; j++) {
        vis[i * p[j]] = true;
        if (i % p[j] == 0) break;
        if (mu[i] == 0)
            mu[i * p[j]] = 0;
        else
            mu[i * p[j]] = mod - mu[i];
    }
}
ci[0] = 1;
for (int i = 1; i <= n; i++) ci[i] = ci[i - 1], ad(ci[i], ci[i - 1]);
for (int i = 1; i <= n; i++)
    if (mu[i])
        for (int j = i; j <= n; j += i) V[j].push_back(i);
for (int i = 1; i <= n; i++) {
    for (int j = 0; j < V[i].size(); j++) ins(V[i][j]);
    Ans[i] = ans;
}
while (T--) read(n), printf("%d\n", Ans[n]);
}
```

C 题：修剪树枝

修剪树枝

输入文件名: tree.in

输出文件名: tree.out

共 20 个测试点，每个测试点 5 分

每个测试点限时 2 秒，运行内存上限 512MB

“阿巴巴，外面那棵树越来越杂乱无章了呢”，老 Win 的同班同学指到。

听到这句话，老 Win 就不自觉羞愧起来，作为一个专业的树枝修剪员，他不允许有人在他面前说出这样的话。

他一拍桌面，跑出操场，一看，原来是最简单的直径树木修剪啊。

这个问题是这样的：

这棵树可以被简化为一棵 n 个点的树，也就是 n 个点 $n - 1$ 条边的无向连通图。

一棵树在今天认为是杂乱无章，当且仅当直径长度 > 老 Win 今天的心情指数。

老 Win 今天做了 10^9 道数竞题，很累，所以他问你，最少切割多少条边才可以使得这棵树切割后所变成的森林中的每一棵树都不杂乱无章。

直径是啥？老 Win 说用尺子量一量就知道。

输入格式

两个正整数 n, k 表示今天老 Win 要处理的树的点数和老 Win 今天的心情指数。

接下来 $n - 1$ 行，每行三个正整数 x, y, c 表示 x 到 y 有一条长度为 c 的边。

输出格式

输出最少切割边数。

数据范围

对于 20% 的数据，满足 $1 \leq n \leq 20$

对于另外 10% 的数据，满足 $y = x + 1$

对于另外 10% 的数据，满足 $x = 1$

对于另外 20% 的数据，满足 $1 \leq n \leq 2000$

对于 100% 的数据，满足 $1 \leq n \leq 1000000, 1 \leq x, y \leq n, x \neq y, 1 \leq c, k \leq 10^9$

样例输入

```
1 5 5
2 1 2 2
3 1 3 3
4 2 4 1
5 2 5 4
```

样例输出

```
1 1
```

样例解释

切割 $(1, 2)$ 这条边最优。

样例输入2

[点击下载](#) (点击“阅读原文”进入信奥题库 NOIP 2020 在线模拟赛下载。)

样例输出2

[点击下载](#) (点击“阅读原文”进入信奥题库 NOIP 2020 在线模拟赛下载。)

题解

考虑贪心。

随便定一个根，往下 dfs 。

返回的时候返回孩子树往下走的最长路径长度。

每次一个点将所有儿子的 dis 排个序，如果最大长度+次大长度 $> k$ ，那么就把最大长度所对应子树的边删掉。

然后就可以得到答案。

考虑贪心的正确性，采用数学归纳法，当只有一个点的时候，啥边都不用删，肯定正确。当前在 x 节点上，任意一个子树的答案都是正确的。

如果最大长度+次大长度 $> k$ ，那么说明最大长度和次大长度所在的这条链上肯定要删一条边，这条边肯定不会删在儿子的子树内，因为儿子的子树已经满足条件了，外面的任意一条链想要进入子树必定要先经过 x 到儿子这两条边，所以删 x 到儿子这两条边严格优于删儿子的子树内的边，而这两条边当然是选择最大长度的链上的那条边，因为两条边删除的代价都是1，当然使最长链消失更好。这个自己细加思考相信一定能理解。

参考标程

```
#include <bits/stdc++.h>

using namespace std;

const int N = 1000010;

struct edge {
    int y, nex, c;
} s[N << 1];

int first[N], len, n, m;
int ans = 0;
void read(int& x)
{
    char ch = getchar();
    x = 0;
    while (ch < '0' || ch > '9') ch = getchar();
    while (ch >= '0' && ch <= '9') x = x * 10 + ch - '0', ch = getchar();
}
void ins(int x, int y, int c)
{
    s[++len] = (edge){y, first[x], c};
    first[x] = len;
}
```

```
int dfs(int x, int fa)
{
    vector<int> V;
    for (int i = first[x]; i != 0; i = s[i].nex)
        if (s[i].y != fa) V.push_back(dfs(s[i].y, x) + s[i].c);
    sort(V.begin(), V.end());

    for (int i = V.size() - 1; i >= 0; i--) {
        if (V[i] > m || i && V[i] > m - V[i - 1])
            ans++;
        else
            return V[i];
    }
    return 0;
}

int main()
{
    freopen("C.in", "r", stdin);
    freopen("C.out", "w", stdout);
    read(n);
    read(m);
    int x, y, c;
    for (int i = 1; i < n; i++) read(x), read(y), read(c), ins(x, y, c), ins(y, x, c);
    dfs(1, 0);
    printf("%d", ans);
}
```

D 题：三元组

三元组

输入文件名: triple.in

输出文件名: triple.out

共 20 个测试点，每个测试点 5 分

每个测试点限时 1 秒，运行内存上限 512MB

“阿巴巴巴，你以为你很牛啊？”老 Win 看着被你切飞的第一题，暗暗自喃道。

老 Win 随便灵机一动，想到了另一道神仙题。

对于三元函数 $f(x, y, z)$ ，当 $z = 0$ 时，有 $f(x, y, z) = 1$ ，当 $z \geq 1$ 时，有 $f(x, y, z) = \sum_{x_1=1}^x \sum_{y_1=1}^y (x - x_1 + 1)(y - y_1 + 1)f(x_1, y_1, z - 1)$

现在给出 N, M, K ，老 Win 想让你告诉他 $\sum_{n=1}^N \sum_{m=1}^M f(n, m, K)$

由于老 Win 最近很讨厌高精度算法，所以他打算直接叫你对 998244353 取模，以减轻蒟蒻出题人的烦恼。

输入格式

多组数据。

第一行一个正整数 T 表示数据组数。

接下来 T 行每行三个正整数表示 N, M, K 。

输出格式

每组数据输出一行一个整数表示答案。

数据范围

对于 20% 的数据，满足 $N, M, K \leq 30$

对于 40% 的数据，满足 $N, M, K \leq 1000$

对于 60% 的数据，满足 $N, M, K \leq 100000$

对于 80% 的数据，满足 $N, M, K \leq 10000000$

对于 100% 的数据，满足 $N, M, K \leq 10^{18}, T \leq 20$

样例输入

| | |
|---|-------------|
| 1 | 2 |
| 2 | 2 3 5 |
| 3 | 342 234 532 |

样例输出

| | |
|---|-----------|
| 1 | 936 |
| 2 | 373362092 |

题解

两道组合数学希望不要被骂。

如果第一步转移错了后面会做的很麻烦，但还是可以用生成函数爆算得到的。

这个转移式子相当于就是在边长为 n, m 的矩形内，选 k 次子矩形的方案数。

那么我们枚举总共选了 x 行和 y 列，那么就相当于将 $2k$ 个没有标号的球放在 x 个盒子里，每个盒子必有球的方案数。

用隔板法简单解决。

$$\sum_{x=1}^n \sum_{y=1}^m C(n, x)C(m, y)C(2k - 1, x - 1)C(2k - 1, y - 1)$$

发现这两个是没有什么关系的，可以解决其中一个。

$$\begin{aligned} & \sum_{x=1}^n C(n, x)C(2k - 1, x - 1) \\ &= \sum_{x=1}^n C(n, n - x)C(2k - 1, x - 1) \\ &= C(n + 2k - 1, n - 1) \end{aligned}$$

答案就是：

$$\begin{aligned} & \sum_{n=1}^N \sum_{m=1}^M C(n + 2K - 1, 2K)C(m + 2K - 1, 2K) \\ &= C(N + 2K, 2K + 1)C(M + 2K, 2K + 1) \end{aligned}$$

首先套用Lucas定理，将每一项都化简成 \mod

发现要处理一个大数阶乘，可以使用快速阶乘算法来踩爆标程。（也许不能

也可以预处理 $(A * 1000000)!$ ，然后每次 $O(1000000)$ 算组合数。

参考标程

```
#include <bits/stdc++.h>
using namespace std;
```

```
int pre[1000010] = {  
    1, 373341033, 45596018, 834980587, 623627864, 428937595, 442819817, 499710224, 833655840, 83857087,  
    295201906, 788488293, 671639287, 849315549, 597398273, 813259672, 732727656, 244038325, 122642896, 310517972,  
    160030060, 483239722, 683879839, 712910418, 384710263, 433880730, 844360005, 513089677, 101492974, 959253371,  
    957629942, 678615452, 34035221, 56734233, 524027922, 31729117, 102311167, 330331487, 8332991, 832392662,  
    545208507, 594075875, 318497156, 859275605, 300738984, 767818091, 864118508, 878131539, 316588744, 812496962,  
    213689172, 584871249, 980836133, 54096741, 417876813, 363266670, 335481797, 730839588, 393495668, 435793297,  
    760025067, 811438469, 720976283, 650770098, 586537547, 117371703, 566486504, 749562308, 708205284, 932912293,  
    939830261, 983699513, 206579820, 301188781, 593164676, 770845925, 247687458, 41047791, 266419267, 937835947,  
    506268060, 6177705, 936268003, 166873118, 443834893, 328979964, 470135404, 954410105, 117565665, 832761782,  
    39806322, 478922755, 394880724, 821825588, 468705875, 512554988, 232240472, 876497899, 356048018, 895187265,  
    808258749, 575505950, 68190615, 939065335, 552199946, 694814243, 385460530, 529769387, 640377761, 916128300,  
    440133909, 362216114, 826373774, 502324157, 457648395, 385510728, 904737188, 78988746, 454565719, 623828097,  
    686156489, 713476044, 63602402, 570334625, 681055904, 222059821, 477211096, 343363294, 833792655, 461853093,  
    741797144, 74731896, 930484262, 268372735, 941222802, 677432735, 474842829, 700451655, 400176109, 697644778,  
    390377694, 790010794, 360642718, 505712943, 946647976, 339045014, 715797300, 251680896, 70091750, 40517433,  
    12629586, 850635539, 110877109, 571935891, 695965747, 634938288, 69072133, 155093216, 749696762, 963086402,  
    544711799, 724471925, 334646013, 574791029, 722417626, 377929821, 743946412, 988034679, 405207112, 18063742,  
    104121967, 638607426, 607304611, 751377777, 35834555, 313632531, 18058363, 656121134, 40763559, 562910912,  
    495867250, 48767038, 210864657, 659137294, 715390025, 865854329, 324322857, 388911184, 286059202, 636456178,  
    421290700, 832276048, 726437551, 526417714, 252522639, 386147469, 674313019, 274769381, 226519400, 272047186,  
    117153405, 712896591, 486826649, 119444874, 338909703, 18536028, 41814114, 245606459, 140617938, 250512392,  
    57084755, 157807456, 261113192, 40258068, 194807105, 325341339, 884328111, 896332013, 880836012, 737358206,  
    202713771, 785454372, 399586250, 485457499, 640827004, 546969497, 749602473, 159788463, 159111724, 218592929,  
    675932866, 314795475, 811539323, 246883213, 696818315, 759880589, 4302336, 353070689, 477909706, 559289160,  
    79781699, 878094972, 840903973, 367416824, 973366814, 848259019, 462421750, 667227759, 897917455, 81800722,  
    956276337, 942686845, 420541799, 417005912, 272641764, 941778993, 217214373, 192220616, 267901132, 50530621,  
    652678397, 354880856, 164289049, 781023184, 105376215, 315094878, 607856504, 733905911, 457743498, 992735713,  
    35212756, 231822660, 276036750, 734558079, 424180850, 433186147, 308380947, 18333316, 12935086, 351491725,  
    655645460, 535812389, 521902115, 67016984, 48682076, 64748124, 489360447, 361275315, 786336279, 805161272,  
    468129309, 645091350, 887284732, 913004502, 358814684, 281295633, 328970139, 395955130, 164840186, 820902807,
```

761699708, 246274415, 592331769, 913846362, 866682684, 600130702, 903837674, 529462989, 90612675, 526540127,
533047427, 110008879, 674279751, 801920753, 645226926, 676886948, 752481486, 474034007, 457790341, 166813684,
287671032, 188118664, 244731384, 404032157, 269766986, 423996017, 182948540, 356801634, 737863144, 652014069,
206068022, 504569410, 919894484, 593398649, 963768176, 882517476, 702523597, 949028249, 128957299, 171997372,
50865043, 20937461, 690959202, 581356488, 369182214, 993580422, 193500140, 540665426, 365786018, 743731625,
144980423, 979536721, 773259009, 617053935, 247670131, 843705280, 30419459, 985463402, 261585206, 237885042,
111276893, 488166208, 137660292, 720784236, 244467770, 26368504, 792857103, 666885724, 670313309, 905683034,
259415897, 512017253, 826265493, 111960112, 633652060, 918048438, 516432938, 386972415, 996212724, 610073831,
444094191, 72480267, 665038087, 11584804, 301029012, 723617861, 113763819, 778259899, 937766095, 535448641,
593907889, 783573565, 673298635, 599533244, 655712590, 173350007, 868198597, 169013813, 585161712, 697502214,
573994984, 285943986, 675831407, 3134056, 965907646, 401920943, 665949756, 236277883, 612745912, 813282113,
892454686, 901222267, 624900982, 927122298, 686321335, 84924870, 927606072, 506664166, 353631992, 165913238,
566073550, 816674343, 864877926, 171259407, 908752311, 874007723, 803597299, 613676466, 880336545, 282280109,
128761001, 58852065, 474075900, 434816091, 364856903, 149123648, 388854780, 314693916, 423183826, 419733481,
888483202, 238933227, 336564048, 757103493, 100189123, 855479832, 51370348, 403061033, 496971759, 831753030,
251718753, 272779384, 683379259, 488844621, 881783783, 659478190, 445719559, 740782647, 546525906, 985524427,
548033568, 333772553, 331916427, 752533273, 730387628, 93829695, 655989476, 930661318, 334885743, 466041862,
428105027, 888238707, 232218076, 769865249, 730641039, 616996159, 231721356, 326973501, 426068899, 722403656,
742756734, 663270261, 364187931, 350431704, 671823672, 633125919, 226166717, 386814657, 237594135, 451479365,
546182474, 119366536, 465211069, 605313606, 728508871, 249619035, 663053607, 900453742, 48293872, 229958401,
62402409, 69570431, 71921532, 960467929, 537087913, 514588945, 513856225, 415497414, 286592050, 645469437,
102052166, 163298189, 873938719, 617583886, 986843080, 962390239, 580971332, 665147020, 88900164, 89866970,
826426395, 616059995, 443012312, 659160562, 229855967, 687413213, 59809521, 398599610, 325666688, 154765991,
159186619, 210830877, 386454418, 84493735, 974220646, 820097297, 2191828, 481459931, 729073424, 551556379,
926316039, 151357011, 808637654, 218058015, 786112034, 850407126, 84202800, 94214098, 30019651, 121701603,
176055335, 865461951, 553631971, 286620803, 984061713, 888573766, 302767023, 977070668, 110954576, 83922475,
51568171, 60949367, 19533020, 510592752, 615419476, 341370469, 912573425, 286207526, 206707897, 384156962,
414163604, 193301813, 749570167, 366933789, 11470970, 600191572, 391667731, 328736286, 30645366, 215162519,
604947226, 236199953, 718439098, 411423177, 803407599, 632441623, 766760224, 263006576, 757681534, 61082578,
681666415, 947466395, 12206799, 659767098, 933746852, 978860867, 59215985, 161179205, 439197472, 259779111,
511621808, 145770512, 882749888, 943124465, 872053396, 631078482, 166861622, 743415395, 772287179, 602427948,
924112080, 385643091, 794973480, 883782693, 869723371, 805963889, 313106351, 262132854, 400034567, 488248149,

265769800, 791715397, 408753255, 468381897, 415812467, 172922144, 64404368, 281500398, 512318142, 288791777,
955559118, 242484726, 536413695, 205340854, 707803527, 576699812, 218525078, 875554190, 46283078, 833841915,
763148293, 807722138, 788080170, 556901372, 150896699, 253151120, 97856807, 918256774, 771557187, 582547026,
472709375, 911615063, 743371401, 641382840, 446540967, 184639537, 157247760, 775930891, 939702814, 499082462,
19536133, 548753627, 593243221, 563850263, 185475971, 687419227, 396799323, 657976136, 864535682, 433009242,
860830935, 33107339, 517661450, 467651311, 812398757, 202133852, 431839017, 709549400, 99643620, 773282878,
290471030, 61134552, 129206504, 929147251, 837008968, 422332597, 353775281, 469563025, 62265336, 835064501,
851685235, 21197005, 264793769, 326416680, 118842991, 84257200, 763248924, 687559609, 150907932, 401832452,
242726978, 766752066, 959173604, 390269102, 992293822, 744816299, 476631694, 177284763, 702429415, 374065901,
169855231, 629007616, 719169602, 564737074, 475119050, 714502830, 40993711, 820235888, 749063595, 239329111,
612759169, 18591377, 419142436, 442202439, 941600951, 158013406, 637073231, 471564060, 447222237, 701248503,
599797734, 577221870, 69656699, 51052704, 6544303, 10958310, 554955500, 943192237, 192526269, 897983911,
961628039, 240232720, 627280533, 710239542, 70255649, 261743865, 228474833, 776408079, 304180483, 63607040,
953297493, 758058902, 395529997, 156010331, 825833840, 539880795, 234683685, 52626619, 751843490, 116909119,
62806842, 574857555, 353417551, 40061330, 822203768, 681051568, 490913702, 9322961, 766631257, 124794668,
37844313, 163524507, 729108319, 490867505, 47035168, 682765157, 53842115, 817965276, 757179922, 339238384,
909741023, 150530547, 158444563, 140949492, 993302799, 551621442, 137578883, 475122706, 443869843, 605400098,
689361523, 769596520, 801661499, 474900284, 586624857, 349960501, 134084537, 650564083, 877097974, 379857427,
887890124, 159436401, 133274277, 986182139, 729720334, 568925901, 459461496, 499309445, 493171177, 460958750,
380694152, 168836226, 840160881, 141116880, 225064950, 109618190, 842341383, 85305729, 759273275, 97369807,
669317759, 766247510, 829017039, 550323884, 261274540, 918239352, 29606025, 870793828, 293683814, 378510746,
367270918, 481292028, 813097823, 798448487, 230791733, 899305835, 504040630, 162510533, 479367951, 275282274,
806951470, 462774647, 56473153, 184659008, 905122161, 664034750, 109726629, 59372704, 325795100, 486860143,
843736533, 924723613, 880348000, 801252478, 616515290, 776142608, 284803450, 583439582, 274826676, 6018349,
377403437, 244041569, 527081707, 544763288, 708818585, 354033051, 904309832, 589922898, 673933870, 682858433,
945260111, 899893421, 515264973, 911685911, 9527148, 239480646, 524126897, 48259065, 578214879, 118677219,
786127243, 869205770, 923276513, 937928886, 802186160, 12198440, 638784295, 34200904, 758925811, 185027790,
80918046, 120604699, 610456697, 573601211, 208296321, 49743354, 653691911, 490750754, 674335312, 887877110,
875880304, 308360096, 414636410, 886100267, 8525751, 636257427, 558338775, 500159951, 696213291, 97268896,
364983542, 937928436, 641582714, 586211304, 345265657, 994704486, 443549763, 207259440, 302122082, 166055224,
623250998, 239642551, 476337075, 283167364, 211328914, 68064804, 950202136, 187552679, 18938709, 646784245,
598764068, 538505481, 610424991, 864445053, 390248689, 278395191, 686098470, 935957187, 868529577, 329970687,

804930040, 84992079, 474569269, 810762228, 573258936, 756464212, 155080225, 286966169, 283614605, 19283401,
24257676, 871831819, 612689791, 846988741, 617120754, 971716517, 979541482, 297910784, 991087897, 783825907,
214821357, 689498189, 405026419, 946731704, 609346370, 707669156, 457703127, 957341187, 980735523, 649367684,
791011898, 82098966, 234729712, 105002711, 130614285, 291032164, 193188049, 363211260, 58108651, 100756444,
954947696, 346032213, 863300806, 36876722, 622610957, 289232396, 667938985, 734886266, 395881057, 417188702,
183092975, 887586469, 83334648, 797819763, 100176902, 781587414, 841864935, 371674670, 18247584};

```

const int mod = 998244353;

int      ksm(int x, int t)
{
    int tot = 1;
    while (t) {
        if (t & 1) tot = 1ll * tot * x % mod;
        x = 1ll * x * x % mod;
        t /= 2;
    }
    return tot;
}

int C(long long x, long long y)
{
    int ans = 1;
    if (x >= mod || y >= mod) ans = C(x / mod, y / mod);
    x %= mod;
    y %= mod;
    if (x < y) return 0;
    int z = x - y, s1 = pre[x / 1000000], s2 = pre[y / 1000000], s3 = pre[z / 1000000];
    for (int j = x / 1000000 * 1000000 + 1; j <= x; j++) s1 = 1ll * s1 * j % mod;
    for (int j = y / 1000000 * 1000000 + 1; j <= y; j++) s2 = 1ll * s2 * j % mod;
    for (int j = z / 1000000 * 1000000 + 1; j <= z; j++) s3 = 1ll * s3 * j % mod;
    return 1ll * ans * s1 % mod * ksm(s2, mod - 2) % mod * ksm(s3, mod - 2) % mod;
}

int main()
{
    freopen("D.in", "r", stdin);
    freopen("D.out", "w", stdout);
    int T;
    scanf("%d", &T);
    long long n, m, k;
    while (T--) {
        scanf("%lld %lld %lld", &n, &m, &k);
        printf("%lld\n", 1ll * C(n + 2 * k, 2 * k + 1) * C(m + 2 * k, 2 * k + 1) % mod);
    }
}

```

NOIP 2020 试题

CCF 全国青少年信息学奥林匹克联赛

CCF NOIP 2020

正式赛

时间：2020 年 12 月 5 日 08:30 ~ 13:00

| 题目名称 | 排水系统 | 字符串匹配 | 移球游戏 | 微信步数 |
|---------|-----------|------------|----------|----------|
| 题目类型 | 传统型 | 传统型 | 传统型 | 传统型 |
| 目录 | water | string | ball | walk |
| 可执行文件名 | water | string | ball | walk |
| 输入文件名 | water.in | string.in | ball.in | walk.in |
| 输出文件名 | water.out | string.out | ball.out | walk.out |
| 每个测试点时限 | 1.0 秒 | 1.0 秒 | 1.0 秒 | 1.0 秒 |
| 内存限制 | 512 MB | 512 MB | 512 MB | 512 MB |
| 子任务数目 | 10 | 25 | 20 | 20 |
| 测试点是否等分 | 是 | 是 | 是 | 是 |

提交源程序文件名

| | | | | |
|--------------|-----------|------------|----------|----------|
| 对于 C++ 语言 | water.cpp | string.cpp | ball.cpp | walk.cpp |
| 对于 C 语言 | water.c | string.c | ball.c | walk.c |
| 对于 Pascal 语言 | water.pas | string.pas | ball.pas | walk.pas |

编译选项

| | |
|--------------|-----|
| 对于 C++ 语言 | -lm |
| 对于 C 语言 | -lm |
| 对于 Pascal 语言 | |

注意事项（请仔细阅读）

1. 文件名（程序名和输入输出文件名）必须使用英文小写。
2. C/C++ 中函数 main() 的返回值类型必须是 int，程序正常结束时的返回值必须是 0。
3. 提交的程序代码文件的放置位置请参照各省的具体要求。
4. 因违反以上三点而出现的错误或问题，申诉时一律不予受理。
5. 若无特殊说明，结果的比较方式为全文比较（过滤行末空格及文末回车）。
6. 程序可使用的栈内存空间限制与题目的内存限制一致。

CCF 全国青少年信息学奥林匹克联赛

正式赛

7. 全国统一评测时采用的机器配置为: Intel(R) Core(TM) i7-8700K CPU @3.70GHz, 内存 32GB。上述时限以此配置为准。
8. 只提供 Linux 格式附加样例文件。
9. 评测在当前最新公布的 NOI Linux 下进行，各语言的编译器版本以其为准。



信息学奥赛指南

排水系统 (water)

【题目描述】

对于一个城市来说，排水系统是极其重要的一个部分。

有一天，小 C 拿到了某座城市排水系统的设计图。排水系统由 n 个排水结点（它们从 $1 \sim n$ 编号）和若干个单向排水管道构成。每一个排水结点有若干个管道用于汇集其他排水结点的污水（简称为该结点的汇集管道），也有若干个管道向其他的排水结点排出污水（简称为该结点的排出管道）。

排水系统的结点中有 m 个污水接收口，它们的编号分别为 $1, 2, \dots, m$ ，污水只能从这些接收口流入排水系统，并且这些结点没有汇集管道。排水系统中还有若干个最终排水口，它们将污水运送到污水处理厂，没有排出管道的结点便可视为一个最终排水口。

现在各个污水接收口分别都接收了 1 吨污水，污水进入每个结点后，会均等地从当前结点的每一个排出管道流向其他排水结点，而最终排水口将把污水排出系统。

现在小 C 想知道，在该城市的排水系统中，每个最终排水口会排出多少污水。该城市的排水系统设计科学，管道不会形成回路，即不会发生污水形成环流的情况。

【输入格式】

从文件 *water.in* 中读入数据。

第一个两个用单个空格分隔的整数 n, m 。分别表示排水结点数与接收口数量。

接下来 n 行，第 i 行用于描述结点 i 的所有排出管道。其中每行第一个整数 d_i 表示其排出管道的数量，接下来 d_i 个用单个空格分隔的整数 a_1, a_2, \dots, a_{d_i} 依次表示管道的目标排水结点。

保证不会出现两条起始结点与目标结点均相同的管道。

【输出格式】

输出到文件 *water.out* 中。

输出若干行，按照编号从小到大的顺序，给出每个最终排水口排出的污水体积。其中体积使用分数形式进行输出，即每行输出两个用单个空格分隔的整数 p, q ，表示排出的污水体积为 $\frac{p}{q}$ 。要求 p 与 q 互素， $q = 1$ 时也需要输出 q 。

【样例 1 输入】

```

1 5 1
2 3 2 3 5
3 2 4 5
4 2 5 4

```

| | |
|---|---|
| 5 | 0 |
| 6 | 0 |

【样例 1 输出】

| | | |
|---|---|---|
| 1 | 1 | 3 |
| 2 | 2 | 3 |

【样例 1 解释】

1 号结点是接收口，4、5 号结点没有排出管道，因此是最终排水口。

1 吨污水流入 1 号结点后，均等地流向 2、3、5 号结点，三个结点各流入 $\frac{1}{3}$ 吨污水。

2 号结点流入的 $\frac{1}{3}$ 吨污水将均等地流向 4、5 号结点，两结点各流入 $\frac{1}{6}$ 吨污水。

3 号结点流入的 $\frac{1}{3}$ 吨污水将均等地流向 4、5 号结点，两结点各流入 $\frac{1}{6}$ 吨污水。

最终，4 号结点排出 $\frac{1}{6} + \frac{1}{6} = \frac{1}{3}$ 吨污水，5 号结点排出 $\frac{1}{3} + \frac{1}{6} + \frac{1}{6} = \frac{2}{3}$ 吨污水。

【样例 2】

见选手目录下的 *water/water2.in* 与 *water/water2.ans*。

【样例 3】

见选手目录下的 *water/water3.in* 与 *water/water3.ans*。

【数据范围】

| 测试点编号 | $n \leq$ | $m \leq$ |
|--------|----------|----------|
| 1 ~ 3 | 10 | |
| 4 ~ 6 | 1000 | 1 |
| 7 ~ 8 | | |
| 9 ~ 10 | 10^5 | 10 |

对于所有测试点，保证 $1 \leq n \leq 10^5$, $1 \leq m \leq 10$, $0 \leq d_i \leq 5$ 。

数据保证，污水在从一个接收口流向一个最终排水口的过程中，不会经过超过 10 个中间排水结点（即接收口和最终排水口不算在内）。

字符串匹配 (string)

【题目描述】

小 C 学习完了字符串匹配的相关内容，现在他正在做一道习题。

对于一个字符串 S ，题目要求他找到 S 的所有具有下列形式的拆分方案数： $S = ABC$, $S = ABABC$, $S = ABAB \cdots ABC$ ，其中 A , B , C 均是非空字符串，且 A 中出现奇数次的字符数量不超过 C 中出现奇数次的字符数量。

更具体地，我们可以定义 AB 表示两个字符串 A, B 相连接，例如 $A = \text{aab}$, $B = \text{ab}$ ，则 $AB = \text{aabab}$ 。

并递归地定义 $A^1 = A$, $A^n = A^{n-1}A$ ($n \geq 2$ 且为正整数)。例如 $A = \text{abb}$ ，则 $A^3 = \text{abbabbabb}$ 。

则小 C 的习题是求 $S = (AB)^iC$ 的方案数，其中 $F(A) \leq F(C)$ ， $F(S)$ 表示字符串 S 中出现奇数次的字符的数量。两种方案不同当且仅当拆分出的 A 、 B 、 C 中有至少一个字符串不同。

小 C 并不会做这道题，只好向你求助，请你帮帮他。

【输入格式】

从文件 *string.in* 中读入数据。

本题有多组数据，输入文件第一行一个正整数 T 表示数据组数。

每组数据仅一行一个字符串 S ，意义见题目描述。 S 仅由英文小写字母构成。

【输出格式】

输出到文件 *string.out* 中。

对于每组数据输出一行一个整数表示答案。

【样例 1 输入】

```

1 3
2 nnrnnr
3 zzaab
4 mmlmmlo

```

【样例 1 输出】

CCF 全国青少年信息学奥林匹克联赛

正式赛 字符串匹配 (string)

```

1 8
2 9
3 16

```

【样例 1 解释】

对于第一组数据，所有的方案为：

1. A=**n**, B=**nr**, C=**nrr**。
2. A=**n**, B=**nrr**, C=**nr**。
3. A=**n**, B=**nrrn**, C=**r**。
4. A=**nn**, B=**r**, C=**nrr**。
5. A=**nn**, B=**rn**, C=**nr**。
6. A=**nn**, B=**rnn**, C=**r**。
7. A=**nrr**, B=**n**, C=**nr**。
8. A=**nrr**, B=**nn**, C=**r**。

【样例 2 输入】

```

1 5
2 kkkkkkkkkkkkkkkkkkkk
3 111111111111rrlllrr
4 cccccccccccccxcxxxcc
5 ccccccccccccaababa
6 ggggggggggggggbaabab

```

【样例 2 输出】

```

1 156
2 138
3 138
4 147
5 194

```

【样例 3】

见选手目录下的 *string/string3.in* 与 *string/string3.ans*。

【样例 4】

见选手目录下的 *string/string4.in* 与 *string/string4.ans*。

【数据范围】

| 测试点编号 | $ S \leq$ | 特殊性质 |
|---------|------------|--------------|
| 1 ~ 4 | 10 | 无 |
| 5 ~ 8 | 100 | |
| 9 ~ 12 | 1000 | |
| 13 ~ 14 | 2^{15} | S 中只包含一种字符 |
| 15 ~ 17 | 2^{16} | S 中只包含两种字符 |
| 18 ~ 21 | 2^{17} | 无 |
| 22 ~ 25 | 2^{20} | |

对于所有测试点，保证 $1 \leq T \leq 5$ ， $1 \leq |S| \leq 2^{20}$ 。

移球游戏 (ball)

【题目描述】

小 C 正在玩一个移球游戏，他面前有 $n + 1$ 根柱子，柱子从 $1 \sim n + 1$ 编号，其中 1 号柱子、2 号柱子、…、 n 号柱子上各有 m 个球，它们自底向上放置在柱子上， $n + 1$ 号柱子上初始时没有球。这 $n \times m$ 个球共有 n 种颜色，每种颜色的球各 m 个。

初始时一根柱子上的球可能是五颜六色的，而小 C 的任务是将所有同种颜色的球移到同一根柱子上，这是唯一的目标，而每种颜色的球最后放置在哪根柱子则没有限制。

小 C 可以通过若干次操作完成这个目标，一次操作能将一个球从一根柱子移到另一根柱子上。更具体地，将 x 号柱子上的球移动到 y 号柱子上的要求为：

1. x 号柱子上至少有一个球；
2. y 号柱子上至多有 $m - 1$ 个球；
3. 只能将 x 号柱子最上方的球移到 y 号柱子的最上方。

小 C 的目标并不难完成，因此他决定给自己加加难度：在完成目标的基础上，使用的操作次数不能超过 820000。换句话说，小 C 需要使用至多 820000 次操作完成目标。

小 C 被难住了，但他相信难不倒你，请你给出一个操作方案完成小 C 的目标。合法的方案可能有多种，你只需要给出任意一种，题目保证一定存在一个合法方案。

【输入格式】

从文件 ***ball.in*** 中读入数据。

第一行两个用空格分隔的整数 n, m 。分别表示球的颜色数、每种颜色球的个数。

接下来 n 行每行 m 个用单个空格分隔的整数，第 i 行的整数按自底向上的顺序依次给出了 i 号柱子上的球的颜色。

【输出格式】

输出到文件 ***ball.out*** 中。

本题采用自定义校验器 (special judge) 评测。

你的输出的第一行应该仅包含单个整数 k ，表示你的方案的操作次数。你应保证 $0 \leq k \leq 820000$ 。

接下来 k 行每行你应输出两个用单个空格分隔的正整数 x, y ，表示这次操作将 x 号柱子最上方的球移动到 y 号柱子最上方。你应保证 $1 \leq x, y \leq n + 1$ 且 $x \neq y$ 。

【样例 1 输入】

```

1 2 3
2 1 1 2
3 2 1 2

```

【样例 1 输出】

```

1 6
2 1 3
3 2 3
4 2 3
5 3 1
6 3 2
7 3 2

```

柱子中的内容为：按自底向上的顺序依次给出柱子上每个球的颜色。

【样例 1 解释】

| 操作 | 1 号柱子 | 2 号柱子 | 3 号柱子 |
|-----|-------|-------|-------|
| 初始 | 1 1 2 | 2 1 2 | |
| 1 3 | 1 1 | 2 1 2 | 2 |
| 2 3 | 1 1 | 2 1 | 2 2 |
| 2 3 | 1 1 | 2 | 2 2 1 |
| 3 1 | 1 1 1 | 2 | 2 2 |
| 3 2 | 1 1 1 | 2 2 | 2 |
| 3 2 | 1 1 1 | 2 2 2 | |

【样例 2】

见选手目录下的 *ball/ball2.in* 与 *ball/ball2.ans*。

【样例 3】

见选手目录下的 *ball/ball3.in* 与 *ball/ball3.ans*。

【数据范围】

| 操作 | 1 号柱子 | 2 号柱子 | 3 号柱子 |
|-----|-------|-------|-------|
| 初始 | 1 1 2 | 2 1 2 | |
| 1 3 | 1 1 | 2 1 2 | 2 |
| 2 3 | 1 1 | 2 1 | 2 2 |
| 2 3 | 1 1 | 2 | 2 2 1 |
| 3 1 | 1 1 1 | 2 | 2 2 |
| 3 2 | 1 1 1 | 2 2 | 2 |
| 3 2 | 1 1 1 | 2 2 2 | |

对于所有测试点，保证 $2 \leq n \leq 50$, $2 \leq m \leq 400$ 。

【校验器】

为了方便选手测试，在`ball` 目录下我们下发了`checker.cpp` 文件，选手可以编译该程序，并使用它校验自己的输出文件。但请注意它与最终评测时所使用的校验器并不完全一致。你也不需要关心其代码的具体内容。

编译命令为：`g++ checker.cpp -o checker`。

`checker` 的使用方式为：`checker <iuputfile> <outputfile>`，参数依次表示输入文件与你的输出文件。

若你输出的数字大小范围不合法，则校验器会给出相应提示。若你的输出数字大小范围正确，但方案错误，则校验器会给出简要的错误信息：

1. `A x`，表示进行到第 x 个操作时不合法。
2. `B x`，表示操作执行完毕后第 x 个柱子上的球不合法。

若你的方案正确，校验器会给出`OK`。

微信步数 (walk)

【题目描述】

小 C 喜欢跑步，并且非常喜欢在微信步数排行榜上刷榜，为此他制定了一个刷微信步数的计划。

他来到了一处空旷的场地，处于该场地中的人可以用 k 维整数坐标 (a_1, a_2, \dots, a_k) 来表示其位置。场地有大小限制，第 i 维的大小为 w_i ，因此处于场地中的人其坐标应满足 $1 \leq a_i \leq w_i$ ($1 \leq i \leq k$)。

小 C 打算在接下来的 $P = w_1 \times w_2 \times \dots \times w_k$ 天中，每天从场地中一个新的位置出发，开始他的刷步数计划（换句话说，他将会从场地中每个位置都出发一次进行计划）。

他的计划非常简单，每天按照事先规定好的路线行进，每天的路线由 n 步移动构成，每一步可以用 c_i 与 d_i 表示：若他当前位于 $(a_1, a_2, \dots, a_{c_i}, \dots, a_k)$ ，则这一步他将会走到 $(a_1, a_2, \dots, a_{c_i} + d_i, \dots, a_k)$ ，其中 $1 \leq c_i \leq k$, $d_i \in \{-1, 1\}$ 。小 C 将会不断重复这个路线，直到他走出了场地的范围才结束一天的计划。（即走完第 n 步后，若小 C 还在场内，他将回到第 1 步从头再走一遍）。

小 C 对自己的速度非常有自信，所以他并不在意具体耗费的时间，他只想知道 P 天之后，他一共刷出了多少步微信步数。请你帮他算一算。

【输入格式】

从文件 ***walk.in*** 中读入数据。

第一行两个用单个空格分隔的整数 n, k 。分别表示路线步数与场地维数。

接下来一行 k 个用单个空格分隔的整数 w_i ，表示场地大小。

接下来 n 行每行两个用单个空格分隔的整数 c_i, d_i ，依次表示每一步的方向，具体意义见题目描述。

【输出格式】

输出到文件 ***walk.out*** 中。

仅一行一个整数表示答案。答案可能很大，你只需要输出其对 $10^9 + 7$ 取模后的值。

若小 C 的计划会使得他在某一天在场地中永远走不出来，则输出一行一个整数 -1。

【样例 1 输入】

```

1 3 2
2 3 3
3 1 1
4 2 -1

```

CCF 全国青少年信息学奥林匹克联赛

正式赛 微信步数 (walk)

5 1 1

【样例 1 输出】

1 21

【样例 1 解释】

从 (1,1) 出发将走 2 步，从 (1,2) 出发将走 4 步，从 (1,3) 出发将走 4 步。

从 (2,1) 出发将走 2 步，从 (2,2) 出发将走 3 步，从 (2,3) 出发将走 3 步。

从 (3,1) 出发将走 1 步，从 (3,2) 出发将走 1 步，从 (3,3) 出发将走 1 步。

共计 21 步。

【样例 2 输入】

```
1 5 4
2 6 8 6 5
3 3 1
4 2 1
5 1 1
6 2 1
7 2 -1
```

【样例 2 输出】

1 10265

【样例 3】

见选手目录下的 *walk/walk3.in* 与 *walk/walk3.ans*。

【样例 4】

见选手目录下的 *walk/walk4.in* 与 *walk/walk4.ans*。

CCF 全国青少年信息学奥林匹克联赛

正式赛 微信步数 (walk)

【数据范围】

| 测试点编号 | $n \leq$ | $k \leq$ | $w_i \leq$ |
|---------|-----------------|----------|------------|
| 1 ~ 3 | 5 | 5 | 3 |
| 4 ~ 6 | 100 | 3 | 10 |
| 7 ~ 8 | 10^5 | 1 | 10^5 |
| 9 ~ 12 | | 2 | 10^6 |
| 13 ~ 16 | 5×10^5 | 10 | |
| 17 ~ 20 | | 3 | 10^9 |

对于所有测试点，保证 $1 \leq n \leq 5 \times 10^5$, $1 \leq k \leq 10$, $1 \leq w_i \leq 10^9$, $d_i \in \{-1, 1\}$ 。