自纳尤坦星而来的非专业级别软件能力认证第一轮（NSP-S）提高级 C++ 语言试题

by x_yi_x

考生注意事项

1. 试题纸共有 0 页，答题纸共有 0 页（因为你用的是电子版），满分 100 分。请不要作答，写在试题纸、答题纸和电子版试题纸上的一律无效。

2. 可以使用任何电子设备（如计算器、手机、充能电弧发射器、粒子光矛等）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. NOIP2019 是第（　）届 NOIP。D
A. 二十四
B. 二十五
C. 九亿九千八百二十四万四千三百五十三
D. 以上皆不对

2. 一个完整的计算机系统应包括（　）C
A. 输入设备和输出设备
B. 集成开发环境和评测系统
C. 硬件系统和软件系统
D. 中央处理器和内存

3. 7 个节点的无标号无根树有（　）个。A
A. 11
B. 12
C. 13
D. 14

4. （　）在 1984 年首次证明了路径压缩且按秩合并的并查集的复杂度是 $\Theta(m\alpha(n))$ 的，其中 m 是操作数，n 是节点数，α 是反阿克曼函数。B
A. Edsger Dijkstra
B. Robert Tarjan
C. Richard Stanley
D. Alan Turing

5. 一棵二叉树的先序遍历为 12345，后序遍历为 24531，则其中序遍历可能为（　）A

A. 21435

B. 21453

C. 12345

D. 24153

6. 下述程序的运行结果是（　）B

```cpp
#include<bits/stdc++.h>
char *s = "#include<bits/stdc++.h>%cchar *s = %c%s%c;%cint main() { printf(s, 10, 34, s, 34, 10); }";
int main() { printf(s, 10, 34, s, 34, 10); }
```

A. 该程序无法运行，因为有编译错误

B.

```cpp
#include<bits/stdc++.h>
char *s = "#include<bits/stdc++.h>%cchar *s = %c%s%c;%cint main() { printf(s, 10, 34, s, 34, 10); }";
int main() { printf(s, 10, 34, s, 34, 10); }
```

C.

```cpp
#include<bits/stdc++.h>
char *s = " ";
int main() { printf(s, 10, 34, s, 34, 10); }
```

D. 以上皆不对

7. 若把序列 $\{5,8,1,3,2,9,4,6,7\}$ 拆分为若干上升子序列，则至少有（　）个上升子序列。B

A. 2

B. 3

C. 4

D. 5

8. 一个 n 个节点的**仙人掌**至多有（　）条边。（仙人掌指的是任何一条边都只出现在一个简单环上的无自环无向图。）B

A. n

B. $2n-2$

C. $2n-1$

D. $2n$

9. 在 C++ 中，表达式 `1LL << 64` 的结果是（　）D

A. 1

B. 0

C. -9223372036854775808

D. 无法确定

10. 队列优化的 Bellman-Ford 算法在最坏情况下的时间复杂度为（　）D

A. $\Theta(n+m)$

B. $\Theta(n^2)$

C. $\Theta((n+m)\log n)$

D. $\Theta(nm)$

11. 下面记 $\oplus$ 为异或，记一个自然数集合的 mex 为其中未出现的最小的自然数。定义一个新运算 $\otimes$：

$$a \otimes b = \mathsf{mex}\{(a' \otimes b) \oplus (a \otimes b') \oplus (a' \otimes b') | 0 \le a' < a, 0 \le b' < b\}$$

则 $4 \otimes 4 =$（　）B

A. 4

B. 6

C. 8

D. 16

12. 若 $T(n) = 2T(n/2) + O(n\log n)$，则 $T(n) =$（　）B

A. $O(n\log n)$

B. $O(n\log^2 n)$

C. $O(n^2)$

D. $O(n^2\log n)$

13. 1, 2, 4, 6 两两异或的最大值是（　）C

A. 5

B. 6

C. 7

D. 8

14. 一个两部各有 n, m 个节点的简单二分图至多有（　）个长度为奇数的简单环。A

A. 0

B. $\max(n, m)$

C. $n+m$

D. nm

15. 下列哪个问题不能直接用贪心解决？（　）C

A. 最小生成树问题

B. 全源最短路问题（边权全为正）

C. 单源最短路问题（边权可能为负）

D. 树上最大独立集问题

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.

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

const int p = 1000000007, maxn = 105;
int n;

struct Z {
    int x;
    Z(int x0 = 0) : x(x0) {}
};

int inline check(int x) { return x >= p ? x - p : x; }
Z operator+(const Z a, const Z b) { return check(a.x + b.x); }
Z operator-(const Z a, const Z b) { return check(a.x - b.x + p); }
Z operator*(const Z a, const Z b) { return 1LL * a.x * b.x % p; }
Z operator-(const Z a) { return check(p - a.x); }
Z &operator+=(Z &a, const Z b) { return a = a + b; }
Z &operator-=(Z &a, const Z b) { return a = a - b; }
Z &operator*=(Z &a, const Z b) { return a = a * b; }

Z fac[maxn], ifac[maxn], inv[maxn];
void init() {
    fac[0] = ifac[0] = fac[1] = ifac[1] = inv[1] = 1;
    for (int i = 2; i < maxn; i++)
        fac[i] = fac[i - 1] * i,
        inv[i] = (p - p / i) * inv[p % i],
        ifac[i] = ifac[i - 1] * inv[i];
}

Z f[maxn][maxn][maxn];

int main(){
    init();
```

```
34     scanf("%d", &n);
35     f[0][1][0] = 1;
36     for (int i = 0; i < n; i++)
37     for (int j0 = 0; j0 <= n - i; j0++)
38     for (int j1 = 0; j1 <= n - i - j0; j1++) if(j0 || j1) {
39         for (int tj0 = 0; tj0 <= n; tj0++)
40         for (int tj1 = 0; tj1 <= n; tj1++)
41             if(j0 - tj0 + tj1 >= 0 && j1 - tj1 + tj0 >= 0)
42                 f[i + j0 + j1][tj0][tj1] += f[i][j0][j1] * ifac[j0 - tj0 + tj1] * ifac[j1 - tj1 + tj0];
43     }
44 
45     printf("%d\n", f[n][0][0] * fac[n]);
46     return 0;
47 }
```

判断题

1) 若输入 3，则该程序输出 12。√

2) 去掉第 38 行的 `if(j0 || j1)`，程序的运行结果与去之前相同。×

3) `fac[13]` 的值为 6227020800。×

4) 只要 $1 \le n \le 100$，`f[][][]` 数组就不可能因为值过大而溢出。√

单选题

5) `inv[71]` 的值为（　）D

A. 284380491

B. 71554167

C. 102279357

D. 98591550

6) 若输入 3，则该程序输出（　）C

A. 60

B. 72

C. 84

D. 108

2.

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 #define itv vector<int>::iterator
```

```

const int maxn = 50005, LEN = 250, maxq = 200005;
int n, q;
int a[maxn];
int K[maxq], ans[maxq];
vector<int> qs[maxn], Y0[LEN], Y1[LEN];

int main() {
    scanf("%d%d", &n, &q);

    for (int i = 1; i <= n; i++)
        scanf("%d", &a[i]);

    for (int i = 1; i <= q; i++) {
        int j;
        scanf("%d%d", &j, &K[i]);
        qs[j].push_back(i);
    }

    for (int i = 1; i <= n; i++) {
        for (int x = a[i], j = 0; j < LEN; j++) {
            if (Y0[j].empty() || Y0[j][Y0[j].size() - 1] >= x) {
                Y0[j].push_back(x);
                break;
            }
            itv pos = upper_bound(Y0[j].begin(), Y0[j].end(), x,
               greater<int>());
            swap(*pos, x);
        }

        for (int x = a[i], j = 0; j < LEN; j++) {
            if (Y1[j].empty() || Y1[j][Y1[j].size() - 1] < x) {
                Y1[j].push_back(x);
                break;
            }
            itv pos = lower_bound(Y1[j].begin(), Y1[j].end(), x);
```

```cpp
            swap(*pos, x);
        }

        for (int id : qs[i]) {
            int k = K[id];
            for (int j = 0; j < LEN && j < k; j++)
                ans[id] += Y0[j].size();
            if (k > LEN)
                for (int j = 0; j < LEN && Y1[j].size() > LEN; j++)
                    ans[id] += min((int)Y1[j].size(), k) - LEN;
        }
    }

    for (int i = 1; i <= q; i++)
        printf("%d\n", ans[i]);
}
```

本题保证输入的 $\{a_1, a_2, ..., a_n\}$ 是 $1 \sim n$ 的一个排列，且 $K_i \geq 1$。

判断题

1) 该程序对询问 (j, K) 的输出 $\leq j$。√

2) 若输入 $\{1, 2, ..., n\}$，则该程序对询问 (j, K) 的输出是 K。×

3) 若输入 $\{n, n-1, ..., 1\}$，则该程序对询问 (j, K) 的输出是 j。√

单选题

4) 该程序对询问 $(j, 1)$ 的输出与（　）相同。D

A. $\{a_1, a_2, ..., a_n\}$ 的最长上升子序列

B. $\{a_1, a_2, ..., a_n\}$ 的最长下降子序列

C. $\{a_1, a_2, ..., a_j\}$ 的最长上升子序列

D. $\{a_1, a_2, ..., a_j\}$ 的最长下降子序列

5) 该程序的时间复杂度为（　）C

A. $\Theta(nq \log n)$

B. $\Theta((n+q)\sqrt{n})$

C. $\Theta(n\sqrt{n}\log n + q\sqrt{n})$

D. $\Theta((n+q)\sqrt{n}\log n)$

6) 该程序的空间复杂度为（　）A

A. $\Theta(n+q)$

B. $\Theta(n\sqrt{n}+q)$

C. $\Theta(n\sqrt{n}\log n + q)$

D. $\Theta(n\sqrt{n}\log n + q\sqrt{n})$

3.

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

const int len = 1000000;
int n[len + 5], k[len + 5], kk[len + 5], nn[len + 5], n_k[len + 5];
char s_[len + 5];
void get(int a[]) {
    scanf("%s", s_); int sl = strlen(s_);
    for (int i = 0; i < sl; i++) a[i] = s_[sl - i - 1] - '0';
}
void del(int a[], int b[], int c[]) {
    int flg = 0;
    for (int i = 0; i < len; i++) {
        c[i] = a[i] - b[i] - flg;
        if (c[i] < 0) c[i] += 2, flg = 1;
        else flg = 0;
    }
    if (flg) cerr << "error!\n";
}

int main() {
    get(n); get(k);
    for (int i = 0; i < len; i++) nn[i] = n[i + 1];
    for (int i = 0; i < len; i++) kk[i] = k[i + 1];
    del(n, k, n_k);

    bool ans1 = 1;
    for (int i = 0; i < len; i++) if (nn[i] < n_k[i]) { ans1 = 0; break; }
    printf("%d\n", ans1);

    del(n, kk, n);
    memset(kk, 0, sizeof(kk)); kk[0] = 1;
    del(n, kk, n);
```

```
    bool ans2 = 1;
    for (int i = 0; i < len; i++) if (n[i] < n_k[i]) { ans2 = 0;
       break; }
    printf("%d\n", ans2);
}
```

判断题

1) 若 n, k 的长度均为 L，则该程序的复杂度为 $\Theta(L)$。√

2) 若输入 `100 10`，则该程序输出 `1 1`。√

3) 即使输入的 $n \geq k$，则该程序仍有可能会输出 `error!`。×

单选题

4) 如果 n 在 $[1, 2^{200}-1]$ 的整数中均匀随机，k 在 $[1, n]$ 的整数中均匀随机，则输出结果最有可能是（ ）A

A. `0 0`

B. `0 1`

C. `1 0`

D. `1 1`

5) `ans1` 的值与以下哪项相等？（ ）C

A. $n-k$ 模 2

B. $\binom{n}{k}$ 模 2

C. 第一类斯特林数的第 n 行第 k 列模 2

D. 第二类斯特林数的第 n 行第 k 列模 2

6) `ans2` 的值与以下哪项相等？（ ）D

A. $n-k$ 模 2

B. $\binom{n}{k}$ 模 2

C. 第一类斯特林数的第 n 行第 k 列模 2

D. 第二类斯特林数的第 n 行第 k 列模 2

三、完善程序（单选题，每小题 3 分，共计 30 分）

1.（叉义叉的结合）叉义叉想在现实生活中体验以撒的结合，于是它打算修建一座这样的地牢：地牢可以被抽象成 n 个点，nm 条边的有向图。1 号点是唯一的入口也是唯一的出口；每一个点恰好有 m 条出边，且这些出边被依次标号为 $[0, m)$ 的正整数；地牢允许自环和重边。

同时，叉义叉希望每一条从 1 号点出发并回到 1 号点的回路都有着一定的规律：具体来说，如果把一条从 1 出发的路径经过的所有边的编号都记录下来，那么能得到一个（可能有前导 0）

的 m 进制数；而对于每一个 m 进制数，自然也就能对应回一条从 1 出发的路径。
于是叉叉叉选定了一个整数 K，它希望这个地牢满足一条从 1 出发的路径能回到 1 当且仅当这条路径对应的数是 K 的倍数。
现在叉叉叉已经选定了 m 和 K，但是它发现并不是对所有的 n，都存在满足上述所有条件的地牢设计方案。建造地牢是一件费时费力的事情，于是叉叉叉想要找到一个最小的满足条件的 n。

```
#include<bits/stdc++.h>
typedef __int128 iint;
using namespace std;

iint m;
iint gcd(iint a, iint b) {
    if (b == 0) return a;
    return ①;
}
iint calc(iint h, iint d, iint k) { //h : 当前可"支配"的点数, d : 将要"损失"的点数
    if (h <= 0) return 0;
    iint g = gcd(m, k), k0 = k / g, m0 = ②;
    if (g == 1) return 0; //无法进行任何合并
    if (h <= k0) return 0; //无法进行任何合并
    d *= ③;
    return h - ④ + calc(k0 - d, d, k0);
}

int main() {
    int T; scanf("%d", &T);
    while (T--) {
        long long m_, k_; scanf("%lld%lld", &m_, &k_);
        m = m_;
        printf("%lld\n", ⑤);
    }
}
```

1) 处应填（ ） C

A. gcd(b, a)

B. gcd(a / b, b)

C. `gcd(b, a % b)`

D. `gcd(a / b, a)`

2) 处应填（　）B

A. `m`

B. `m / g`

C. `k0 / g`

D. `m * k0 / g`

3) 处应填（　）D

A. `k`

B. `k0`

C. `m`

D. `m0`

4) 处应填（　）B

A. `k`

B. `k0`

C. `m`

D. `m0`

5) 处应填（　）D

A. `calc(k_ - 1, 1, k_)`

B. `(long long)calc(k_ - 1, 1, k_)`

C. `(long long)calc(k_ - 1, 1, k_) - k_`

D. `k_ - (long long)calc(k_ - 1, 1, k_)`

2.（好吃的题目）有一条小吃街，从左到右依次排列着 n 个商店，从 1 开始标号。第 i 个商店会只出售一种小吃，热量为 h_i，美味度为 w_i。

现在有 m 个吃货要来逛街，第 i 个吃货会在 $[l_i, r_i]$ 的商店内寻找小吃，而且为了防止太胖，最多能摄入 t_i 的热量。小吃吃多了会腻，所以同一个商店的小吃只能吃一次。

现在每个吃货想知道自己最多能得到多少美味度。

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

const int maxn = 65536;
int n, q;
int V[maxn], W[maxn];
vector<vector<int> > FL, FR;
vector<int> emp;
int Lg2[maxn];
```

```
vector<int> Qs[maxn];
int ANS[200005], L[200005], R[200005], T[200005];
void maxeq(int &x, int y) { x = max(x, y); }

void init() {
    emp.resize(201);
    Lg2[0] = 0;
    for (int i = 1; i < 65536; i++) Lg2[i] = Lg2[①] + 1;
}

int Merge(vector<int> const T1, vector<int> const T2, int S) {
    int ans = 0;
    for (int i = 0; i <= S; i++) maxeq(ans, T1[i] + T2[S - i]);
    return ans;
}

void Solve(int x, int l, int r) {
    if (l == r) return;
    int mid = (l + r) >> 1;
    ②;
    for (int i = mid; i >= l; i--) {
        vector<int> tmp = emp;
        for (int j = 0; j <= 200; j++) {
            if (FL.size()) tmp[j] = FL.back()[j];
            if (j >= V[i])
                if (FL.size()) maxeq(tmp[j], FL.back()[j - V[i]] +
                   W[i]);
                else maxeq(tmp[j], W[i]);
        }
        FL.push_back(tmp);
    }
    FR.clear();
    for (int i = mid + 1; i <= r; i++) {
        vector<int> tmp = emp;
        for (int j = 0; j <= 200; j++) {
            if (FR.size()) tmp[j] = FR.back()[j];
```

```cpp
            if (j >= V[i])
                if (FR.size()) maxeq(tmp[j], FR.back()[j - V[i]] + W[i]);
                else maxeq(tmp[j], W[i]);
        }
        FR.push_back(tmp);
    }
    for (int id : Qs[x]) ANS[id] = Merge(FL[mid - L[id]], FR[③], T[id]);
    Solve(x << 1, l, mid), Solve(x << 1 | 1, mid + 1, r);
}

int LCA(int x, int y) {return (x + n) >> Lg2[x ④ y];}

int main() {
    init();

    scanf("%d%d", &n, &q);
    for (int i = 0; i < n; i++) scanf("%d", &V[i]);
    for (int i = 0; i < n; i++) scanf("%d", &W[i]);

    int N0 = 1;
    while (N0 < n) ⑤;
    n = N0;
    for (int i = 1; i <= q; i++) {
        scanf("%d%d%d", &L[i], &R[i], &T[i]); L[i]--, R[i]--;
        if (L[i] == R[i]) {
            ANS[i] = (V[L[i]] <= T[i] ? W[L[i]] : 0);
            continue;
        }
        Qs[LCA(L[i], R[i])].push_back(i);
    }
    Solve(1, 0, n - 1);
    for (int i = 1; i <= q; i++) printf("%d\n", ANS[i]);
}
```

1) ①处应填（ ）C

A. `i - 1`

B. `i % 2`

C. `i / 2`

D. `i * 2`

2) 处应填（ ）C

A. `init()`

B. `emp.resize(r - l + 1);`

C. `FL.clear()`

D. `FR.clear()`

3) 处应填（ ）D

A. `mid + 1 - R[id]`

B. `mid - R[id]`

C. `R[id] - mid`

D. `R[id] - mid - 1`

4) 处应填（ ）C

A. `|`

B. `&`

C. `^`

D. `%`

5) 处应填（ ）D

A. `NO++`

B. `n--`

C. `n >>= 1`

D. `NO <<= 1`