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

```
#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.

```
#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.

```
#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. $O(n^2 \log n)$

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）

3.

```
#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 $[1, 2^200-1]$ 中随机，k 在 $[1, n]$ 中随机，则答案最有可能是（　）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）

2.（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。

于是叉义叉选定了一个整数 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。

现在叉义叉已经选定了 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，于是叉义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 壹;
}
```

```
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 = 贰;
    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 *= 叁;
    return h - 肆 + 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", 伍);
    }
}
```

1) 壹处应填（ ）C

A. gcd(b, a)

B. gcd(a / b, b)

C. gcd(b, a % b)

D. gcd(a / b, a)

2) 贰处应填（ ）B

A. m

B. m / g

C. k0 / g

D. m * k0 / g

3) 叁处应填（ ）D

A. k

B. k0

C. m

D. m0

4) 肆处应填（ ）B

A. k

B. k0

C. `m`

D. `m0`

5) 伍处应填（　）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_)