YDSP Senior 组模拟考试

本次考试的答题时间为两个小时，请各位选手自行掌握时间。
提交请访问 yundouxueyuan.com 参加对应的比赛，将最终答案以程序的形式提交至对应题目。
更进一步的提交细节，请参考对应题目里的说明。

最后，本次模拟考试为纯公益目的，不得用于任何未经授权的盈利活动。

* 祝考试顺利 *

云斗学院
Yun Dou Xue Yuan

2025 云斗学院软件能力认证第一轮
(YDSP - Senior) 提高级 C++ 语言试题

考生注意事项:

- 试题纸一共 15 页，满分 100 分。作答后请记录答案，并按要求提交至对应比赛处。
- 考试过程中不得使用任何电子设备或查阅任何书籍资料。

1 选择题（每题 2 分，共 30 分）

1.1 第 1 题

线性同余方程组 $\begin{cases} x \equiv -1 \pmod 7 \\ x \equiv 9 \pmod{17} \end{cases}$ 的最小自然数解的个位是 __▲__。

A. 1
B. 3
C. 6
D. 8

1.2 第 2 题

你需要维护一个初始为空的**多重集合** S，支持“添加一个数 x”和“询问 x 在 S 中出现的次数”。下列 STL 容器中，能保证单次操作最坏复杂度为 $O(\log |S|)$ 的前提下，使用最方便的是 __▲__。

A. `set`
B. `multiset`
C. `priority_queue`
D. `map`

1.3 第 3 题

在 Linux 系统中，返回上一级目录的命令是 __▲__。

A. `back`
B. `cd /.`
C. `cd`
D. `cd ..`

1.4 第 4 题

一张 100 个结点的简单无向图存在欧拉路径，则图上至多有 __▲__ 条边。

A. 4950
B. 4900
C. 99
D. 前三个选项都不对

1.5 第 5 题

以下程序片段的输出是 ___▲___。

```
bitset<8> a;
a[4]=1; a.flip(); a.flip(2);
a&=a<<3; cout<<a;
```

A. 01001000

B. 00010010

C. 00001001

D. 01001111

1.6 第 6 题

Alice 给一些长度为 200 的 01 串进行自然溢出哈希。具体地，串 $a_0a_1\ldots a_k$ 的哈希值为 $(\sum_{i=0}^{k} a_iB^i) \bmod (2^{64})$ 。下列 B 的取值中，最合适的是 ___▲___。

A. 1

B. 6

C. 27

D. $2^{63}+1$

1.7 第 7 题

一棵二叉树满足小根堆的性质，其中序遍历为 $9,1,4,6,2,8,7,3,5$，则其前序遍历为 ___▲___。

A. $9,8,6,4,1,2,7,5,3$

B. $1,2,4,6,3,5,7,8,9$

C. $1,4,2,6,5,3,7,8,9$

D. $1,4,2,6,3,5,7,8,9$

1.8 第 8 题

在 n 个结点、m 条边的稀疏（$m=\Theta(n)$）有向图中使用 SPFA 算法求 1 结点到所有结点的最短路，保证没有负环。如果使用优先队列替换原来的队列，那么 ___▲___。

A. 它的最坏时间复杂度为 $O(nm)$。

B. 正权图中，它的时间复杂度为 $O(n\log n)$。

C. 在有负边权时，算法可能无法结束。

D. 令 $T(n)$ 为 n 个结点的强连通图对应的最小用时，则 $T(n)=O(n\log n)$。

1.9 第 9 题

现有一棵 2025 个结点的有根树，根为 1，满足结点 $i(2\le i\le 2025)$ 的父结点为 $\lfloor i/2\rfloor$。

通过更改根结点，可以让结点 920 和 457 的最近公共祖先变成结点 ___▲___。

A. 28

B. 914

C. 460

D. 231

1.10 第 10 题

定义递归函数 $f(x)=\begin{cases}1 & x\geq 2\\ 0 & 1<x<2\\ f(3x) & x\leq 1\end{cases}$。若 x 是在 $\frac{1}{6}$ 和 $\frac{1}{2}$ 之间随机均匀选取的实数，那么 $f(x)=1$ 的概率是 __▲__。

A. $\frac{1}{9}$

B. $\frac{1}{6}$

C. $\frac{1}{3}$

D. 前三个选项都不对

1.11 第 11 题

8 个小朋友手拉手围成了两个圆圈，每个圆圈有 4 人，所有小朋友都面向圆圈内部。两个圆圈没有顺序，且圆圈旋转后重合的方案视为一种。他们共有 __▲__ 种排法。

A. 1260

B. 2520

C. 5040

D. 前三个选项都不对

1.12 第 12 题

一程序的运行时间 $T(n)$ 满足 $T(1)=O(1)$，$T(n)=T(\lfloor\sqrt{n}\rfloor)+\log_2(n)$，那么其时间复杂度为 __▲__。

A. $O(\log n)$

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

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

D. $O(\sqrt{n})$

1.13 第 13 题

针对普通快速排序在数组基本有序时的时间复杂度退化问题，我们把给 $a[l,\ldots,r]$ 排序时的基准值选取过程改成“取 `a[l],a[r],a[(l+r)/2]` 的中位数”。然而这个新算法仍然有可能退化成 $O(n^2)$，例如 __▲__。

A. $2,3,4,\ldots,n,1,n+1,n+2,\ldots,2n$

B. $1,3,5,\ldots,2n-1,2n,2n-2,\ldots,6,4,2$

C. $1,2,\ldots,2n$ 随机打乱后形成的序列

D. 前三个选项都不对

1.14 第 14 题

在 3^{48}，$\varphi(119)$，C_9 中，模 97 意义下和 96! 同余的数有 __▲__ 个。（注：C_n 为卡特兰数，$C_8=1430$ 。）

A. 0

B. 1
C. 2
D. 3

1.15 第 15 题

NOI 2025 在 7 月 18 日圆满落幕。这次比赛是 NOI 全国赛第三次落地历史文化名城 __▲__，信息学领域的青年才俊在烟雨时节齐聚江南水乡，共赴这场代表中国青少年信息学最高水平的盛会。

A. 临沂
B. 绍兴
C. 杭州
D. 南京

2 阅读程序（无特殊说明时判断 1.5 分，选择 3 分，共 40 分）

2.1 第 1 题（12 分）

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

long long x, y;

long long function_1(long long u) {
    return u ^ ((1ull << 63) - 1);
}

long long function_2(long long u, long long v) {
    return u & function_1(v);
}

long long function_3(long long u, long long v) {
    return u & v;
}

int main() {
    cin >> x >> y;
    cout << (x & y) << "\n";
}
```

所有输入均在 $[0, 2^{63} - 1]$ 范围内。上述代码在运行下述题目时均开启 O2 优化。

2.1.1 判断题

1. 若将第 21 行更换为 `cout << (function_2(x, y) & function_2 (y, x)) << "\n";` 后程序输出结果不变。

2. 若将第 7 行替换为 `return u ^ (1ull << 63) - 1;`后结果不变。

3. (2 分)若将第 21 行更换为 `cout << (function_2 (x, y) | function_2 (y, x)) << "\n";` 后，输出的结果会大于等于任意一个输入的数。

2.1.2 选择题

1. 若将第 4 行替换为 `unsigned int x, y;`，将第 21 行替换为 `cout << (function_2(x, y) | function_2(y, x) | function_3(x, y)) << "\n";` 且输入为 12345678910 10987654321，那么输出结果为 ___▲___。

   A. 10985409584

   B. 12347923647

   C. 4294967293

   D. 以上结果均可能不正确。

2. (4 分) 当输入为 68 53 时，将第 21 行替换为 ___▲___ 时，输出结果最大。

   A. `cout << (function_2(x, y) | function_3(x, y)) << "\n";`

   B. `cout << (function_2(y, x) | function_3(x, y)) << "\n";`

   C. `cout << (function_2(x, y) & function_3(x, y)) << "\n";`

   D. `cout << (function_2(y, x) & function_3(x, y)) << "\n";`

2.2 第 17 题 (14 分)

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

const int N = 1.01e6;
char s[N];
int n, l[N], r[N];
int f[N][2];

int solve1() {
    int res = 0;

    for (int i = 1, x = 0; i <= n; i++) {
        l[i] = l[i - 1];

        if (s[i] == '0')
            l[i] += x;
        else ++x, res += l[i - 1] + 1;
    }
```

```
20
21      for (int i = n, x = 0; i >= 1; i--) {
22          r[i] = r[i + 1];
23
24          if (s[i] == '0')
25              r[i] += x;
26          else ++x;
27      }
28
29      for (int i = 1; i <= n; i++) {
30          if (s[i] == '1')
31              res += l[i - 1] * r[i + 1];
32      }
33
34      return res;
35  }
36
37  int solve2() {
38      for (int i = 1; i <= n; i++) {
39          f[i][0] = f[i - 1][0];
40          f[i][1] = f[i - 1][1];
41
42          if (s[i] == '0') {
43              f[i][0] += f[i - 1][1];
44          } else {
45              f[i][1] += f[i - 1][0] + 1;
46          }
47      }
48
49      return f[n][1];
50  }
51
52  int main() {
53      cin >> n;
54
55      if (n > 50) {
56          printf("0 1\n");
57          return 0;
58      }
59
60      scanf("%s", s + 1);
61      int ans1 = solve1();
```

```
    int ans2 = solve2();
    printf("%d %d\n", ans1, ans2);
    return 0;
}
```

假设输入的 s 是包含 n 个字符的 01 串，完成下面的判断题和单选题。

2.2.1 判断题

1. 输入 7 1001101 时，程序输出两个数 15 和 15。
2. 无论任何合法输入，solve1() 的返回值总是不超过 solve2() 的返回值。
3. （2 分）当 $n = 10$ 时存在至少一种输入，使得 solve1() 的返回值不小于 50。

2.2.2 选择题

1. （2 分）如果程序输出的两个数不同，n 的最小值为 __▲__。

   A. 6

   B. 7

   C. 8

   D. 51

2. 当 $n = 10$ 时，有 __▲__ 种合法的输入使得程序输出的两个数字相同。

   A. 648

   B. 720

   C. 848

   D. 1024

3. （4 分）当程序输出的两个数相差为 128 时，输入可能是 __▲__。

   A. 14 11001011010011

   B. 14 10101000010101

   C. 14 10111000011101

   D. 14 11001100110011

2.3 第 18 题（14 分）

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

int n, m, dp[1 << 20];
int ty, a[20];
int pc[1 << 20];
```

```
7
8   int main() {
9       cin >> n >> m;
10      cin >> ty;
11
12      if (ty == 1) {
13          for (int i = 0; i < n; i++)
14              a[i] = i;
15      } else {
16          for (int i = 0; i < n; i++)
17              cin >> a[i];
18      }
19
20      dp[0] = 1;
21
22      for (int i = 0; i < (1 << 20); i++) {
23          for (int t = 0; t < 20; t++) {
24              if (i & (1 << t))
25                  ++pc[i];
26          }
27      }
28
29      for (int i = 0; i < (1 << n); i++) {
30          for (int j = i; j; j = (j - 1) & i) {
31              int x = 0;
32
33              for (int t = 0; t < n; t++) {
34                  if (j & (1 << t))
35                      x ^= a[t];
36              }
37
38              if (pc[x] == m)
39                  dp[i] |= dp[i - j];
40          }
41      }
42
43      if (dp[(1 << n) - 1])
44          printf("Yes\n");
45      else
46          printf("No\n");
47
48      return 0;
```

```
49 }
```

输入的 n 为不超过 20 的正整数。

2.3.1 判断题

1. 当输入的 $ty = 1$ 时，将第 21 行的 $t < 20$ 改为 $t < 5$ ，输出不变。
2. 输入 5 2 1 时，程序输出 Yes。
3. （2 分）当输入的 $n = 16$，$ty = 1$ 且 $0 \le m \le 4$ 时，程序总是会输出 Yes。

2.3.2 选择题

1. （2 分）当输入的前三个数字满足 $n = 4, m = 1, ty = 2$ 时，程序会继续输入 4 个数字，输入 __▲__ 会使程序输出 No。

   A. 2 4 16 256

   B. 5 6 7 8

   C. 17 21 5 7

   D. 3 7 12 48

2. 下列描述正确的是 __▲__。

   A. 当输入的 n 为偶数，$m = 1, ty = 1$ 的时候，程序输出 Yes。

   B. 当输入的 n 为奇数，$m = 1, ty = 1$ 的时候，程序输出 Yes。

   C. 当输入的 n 为偶数，$m = 0, ty = 1$ 的时候，程序输出 Yes。

   D. 当输入的 n 为奇数，$m = 0, ty = 1$ 的时候，程序输出 Yes。

3. （4 分）该程序的时间复杂度为 __▲__。

   A. $O(n)$

   B. $O(n2^n)$

   C. $O(n3^n)$

   D. $O(n4^n)$

3 完善程序（单选题，每小题 3 分，共计 30 分）

3.1 Genshin（15 分）

现在是 2030 年，MHY 已经打造出全球十亿人愿意生活在其中的虚拟世界，所以你需要维护一个长度为 n，由且仅由 A,B,C 构成的序列 a，你需要支持以下操作：

- A l r c：把 $a_l, a_{l+1}, ..., a_r$ 全部改成 $c (c \in \{A, B, C\})$；
- B l r：询问 $a_l, a_{l+1}, ..., a_r$ 是否完全相同，并且 $a_{l-1} \neq a_{r+1}$。

```cpp
#include <bits/stdc++.h>
using namespace std;
struct odt_node {
    int l, r;
    mutable char val;
    odt_node(int a = 0, int b = 0, char c = 0): l(a), r(b), val(c) {}
    bool operator<(const odt_node &t)const {
        return (/* Blank 1 */);
    }
};
int n, q, pre = 1;
char c1, c2;
struct odt: (/* Blank 2 */) {
    auto split(int p) {
        auto it = (/* Blank 3 */)(p);

        if (it != end() && it->l == p)
            return it;

        auto t = *--it;
        erase(it);
        emplace(t.l, p - 1, t.val);
        return emplace(p, t.r, t.val).first;
    }
    auto assign(int l, int r, char k) {
        erase(split(l), split(r + 1));
        return emplace(l, r, k).first;
    }
    bool query(int l, int r) {
        auto ir = split(r + 1), il = split(l);

        if (l != 1 && r != n && prev(il)->val == ir->val)
            return false;

        for (auto it = il; it != ir; ++it)
            if (il->val != it->val)
                return assign(il->l, (/* Blank 4 */), il->val), false;

        return assign(l, r, il->val), true;
    }
}
s;
```

```
signed main() {
    cin >> n >> c2;

    for (int i = 2; i <= n; i++) {
        cin >> c1;

        if (c1 != c2)
            s.emplace(pre, i - 1, c2), pre = i;

        c2 = c1;
    }

    (/* Blank 5 */);
    s.emplace(n + 1, n + 1, 'D');
    cin >> q;

    for (int l, r; q--;) {
        char op, k;
        cin >> op;

        if (op == 'A')
            cin >> l >> r >> k, s.assign(l, r, k);
        else
            cin >> l >> r, cout << (s.query(l, r) ? "Yes\n" : "No\n");
    }

    return 0;
}
```

1. /* Blank 1 */ 处应该填 __▲__。

   A. `l < t.l`

   B. `t.l < l`

   C. `r < t.l`

   D. `t.r < r`

2. /* Blank 2 */ 处应该填 __▲__。

   A. `vector<odt_node>`

   B. `list<odt_node>`

   C. `set<odt_node>`

   D. `map<odt_node>`

3. /* Blank 3 */ 处应该填 __▲__。

A. `lower_bound`

B. `upper_bound`

C. `find`

D. `count`

4. /* Blank 4 */ 处应该填 __▲__。

A. `(--it)->r`

B. `(it--)->r`

C. `(++it)->r`

D. `(it++)->r`

5. /* Blank 5 */ 处应该填 __▲__。

A. `s.emplace(1,n,c1)`

B. `s.emplace(1,n,c2)`

C. `s.emplace(pre,n,c1)`

D. `s.emplace(pre,n,c2)`

3.2 Impact（15 分）

正在拯救世界的技术宅定义一个字符串 T 是好的，当且仅当存在字符串 T' 和整数 $k(k \geq 2)$，使得 T 可以由 k 个 T' 首尾相接得到。

例如，字符串 `gogogo` 就是好的，因为它可以由 3 个字符串 `go` 首尾相接得到；而 `power` 就不是好的。

给定仅包含小写英文字母的字符串 $S(|S| \leq 80)$。你需要求出 S 最长的好的子序列的长度，特别的如果 S 没有好的子序列，答案为 0。

```
#include <bits/stdc++.h>
#define siz(x) int((x).size())
using namespace std;
template<typename T1, typename T2>
T1 &cmin(T1 &x, T2 &&y) {
    if (y < x)
        x = y;

    return x;
}

template<typename T1, typename T2>
T1 &cmax(T1 &x, T2 &&y) {
    if (x < y)
        x = y;

```

```
    return x;
}


int n, ans, B;
string s;
void solve(const string &a, const string &b) {
    int f[siz(a) + 1][siz(b) + 1];
    memset(f, 0, sizeof f);

    for (int i = 1; i <= siz(a); i++)
        for (int j = 1; j <= siz(b); j++) {
            f[i][j] = max(f[i - 1][j], f[i][j - 1]);

            if (a[i - 1] == b[j - 1])
                cmax(f[i][j], f[i - 1][j - 1] + 1);
        }

    cmax(ans, 2 * f[siz(a)][siz(b)]);
}
void solve(const string &a, const string &b, const string &c) {
    int f[siz(a) + 1][siz(b) + 1][siz(c) + 1];
    memset(f, 0, sizeof f);

    for (int i = 1; i <= siz(a); i++)
        for (int j = 1; j <= siz(b); j++)
            for (int k = 1; k <= siz(c); k++) {
                f[i][j][k] = max({f[i - 1][j][k], f[i][j - 1][k], f[i][j][k - 1]});

                if (/* Blank 1 */)
                    cmax(f[i][j][k], f[i - 1][j - 1][k - 1] + 1);
            }

    cmax(ans, 3 * f[siz(a)][siz(b)][siz(c)]);
}


signed main() {
    cin >> s;
    n = siz(s);
    B = (/* Blank 2 */);

```

```
    for (int i = 1; i < n; i++)
        solve({&s[0], &s[i]}, {&s[i], &s[n]});

    for (int i = 1; i < n; i++)
        for (int j = i + 1; j < n; j++)
            solve({&s[0], &s[i]}, {&s[i], &s[j]}, {&s[j], &s[n]});

    for (int l = 0; l < n; l += B) {
        int r = min(n - 1, l + B - 1), m = r - l + 1;

        for (int k = 1; k < (/* Blank 3 */); k++) {
            string t;
            int L = n, R = 0, res = 1;

            for (int i = 0; i < m; i++)
                if (k >> i & 1)
                    t += (/* Blank 4 */), cmin(L, l + i), cmax(R, l + i);

            int p = 0;

            for (int i = 0; i < L; i++)
                if (t[p] == s[i])
                    res += (/* Blank 5 */);

            p = 0;

            for (int i = R + 1; i < n; i++)
                if (t[p] == s[i])
                    res += (/* Blank 5 */);

            if (res > 1)
                cmax(ans, res * siz(t));
        }
    }

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

1. /* Blank 1 */ 处应该填 __▲__。

   A. a[i-1]==b[j-1]

   B. a[i-1]!=b[j-1]

C. `a[i-1]==b[j-1]&&b[j-1]==c[k-1]`

D. `a[i-1]==b[j-1]||b[j-1]==c[k-1]`

2. /* Blank 2 */ 处应该填 __▲__。

A. `4`

B. `8`

C. `16`

D. `32`

3. /* Blank 3 */ 处应该填 __▲__。

A. `(1<<(m-1))-1`

B. `(1<<(m-1))`

C. `(1<<m)-1`

D. `(1<<m)`

4. /* Blank 4 */ 处应该填 __▲__。

A. `s[l-i]`

B. `s[l+i]`

C. `s[r-i]`

D. `s[r+i]`

5. /* Blank 5 */ 处应该填 __▲__。

A. `!((++p)%=siz(t))`

B. `!((p++)%=siz(t))`

C. `!!((++p)%=siz(t))`

D. `!!((p++)%=siz(t))`