YDSP Junior 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 - Junior) 入门级 C++ 语言试题

考生注意事项:

- 试题纸一共 14 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

报名 CSP-J 第一轮的选手，需要在当年的 9 月 1 日满 __▲__ 周岁。

A. 8

B. 10

C. 12

D. 14

1.2 第 2 题

下面四条 C++ 语句中，__▲__ **不是**正确的声明变量语句。

A. `long long ago;`

B. `double kill;`

C. `int eger;`

D. `define the,world;`

1.3 第 3 题

Alice 使用 C++ 编写了一款小程序，代码文件为 `game.cpp`，并使用软件 `DEV-C++` 生成了可执行文件 `game.exe`，并创建了快捷方式 井字棋。这天，Bob 想要用 U 盘拷走 Alice 的游戏。Alice 应该把 __▲__ 拷进 Bob 的 U 盘，才能让 Bob 直接游玩。

A. `game.cpp`

B. `DEV-C++`

C. `game.exe`

D. 井字棋

1.4 第 4 题

现在有 `int` 类型的变量 x, 保证 $0 \le x \le 100$, 现要判断一个自然数 x 是不是 8 的倍数，下列 C++ 表达式正确的是 __▲__。

A. `x&7==0`

B. `(x&-x)>7`

C. `(x|7)==0`

D. `x%(2^3)==0`

1.5 第 5 题

考虑两个十进制正整数 x, y，其中 x 是 n 位数，y 是 m 位数，$x \geq y$。那么 ___▲___。

A. x, y 的最高位如果不同，那么 x 的最高位数值更大。

B. $x + y$ 是 n 位数或 $n + 1$ 位数。

C. $x - y$ 是 n 位数或 $n - 1$ 位数。

D. xy 是 nm 位数。

1.6 第 6 题

一棵二叉树的前序遍历是 ECDAHBIGF，中序遍历是 DCHBIAEGF，则其后序遍历是 ___▲___。

A. DIBHACFGE

B. DHBIACGFE

C. GFAHBIDCE

D. EGFCAHBID

1.7 第 7 题

我们知道，01 背包问题是指，有 n 个物品，每个物品有重量 w_i 和价值 v_i，要选出一部分物品使得重量和不超过给定值 W，然后最大化选出物品的价值和。一个常见错误做法是贪心地按照物品性价比从高到低排序，然后从前往后装入背包，直到无法装入下一个物品。事实上，当 $n = 2, W = 10$，$w_1 = 5, v_1 = 10$ 时，只要令 ___▲___ 就可以让这个做法得到错误结果。

A. $w_2 = 3, v_2 = 8$

B. $w_2 = 6, v_2 = 13$

C. $w_2 = 8, v_2 = 18$

D. $w_2 = 7, v_2 = 12$

1.8 第 8 题

Alice 和 Bob 正在复习 CSP-J 知识点。

Alice: “一个栈可以用 _____ 实现，且可以做到单个元素入栈、出栈均为 $O(1)$。”

Bob: “不对，如果用这个实现，入栈 $O(1)$，出栈就做不到 $O(1)$。“

已知 Bob 这句话是对的，判断横线处应该填 ___▲___。

A. `std::vector`

B. `std::list`

C. `std::stack`

D. `std::queue`

1.9 第 9 题

现有一个 n 项的数组 `a[1],...,a[n]`，保证 `a[0]` 等于 0。

- 如果要原地求 `a` 的前缀和，应该写 `for(/* Blank 1 */)a[i]+=a[i-1];`
- 如果要原地求 `a` 的差分，应该写 `for(/* Blank 2 */)a[i]-=a[i-1];`

给出正向循环 `int i=1;i<=n;i++` 和反向循环 `int i=n;i>0;i--`。序号处应分别填写 ___▲___。

A. 正向循环、正向循环

B. 正向循环、反向循环
C. 反向循环、正向循环
D. 反向循环、反向循环

1.10 第 10 题

我们规定，对于一个三位数（可以有前导 0，即 000 ~ 999），如果其含有数码 2, 3, 4, 5, 7，那么其**不可倒置**，否则其倒置的结果可以通过以下两步得出：

- 把数码 9 变成 6，把数码 6 变成 9，**同时进行**。
- 交换三位数的百位和个位。

例如 089 倒置的结果就是 680。

如果一个三位数不可倒置或者倒置结果和原来的三位数相同，就称这个三位数是安全的。那么，一共有 ▲ 个安全的三位数。

A. 890
B. 875
C. 900
D. 925

1.11 第 11 题

计算 $(2025)_8 + (920)_{10}$ 的结果是 ▲。

A. $(11110101101)_2$
B. $(3665)_8$
C. $(1935)_{10}$
D. $(7AB)_{16}$

1.12 第 12 题

后缀表达式 `2 4 * 7 - 1 2 + *` 的计算结果为 ▲。

A. 1
B. 2
C. 3
D. 4

1.13 第 13 题

对序列 2, 0, 2, 5, 0, 9, 2, 0 进行升序冒泡排序的过程中，一共发生了 ▲ 次“交换第 4 个和第 5 个数”。

A. 1
B. 2
C. 3
D. 4

1.14 第 14 题

不能写成两个合数之和的最大自然数 n 是 ▲ 。

A. 2

B. 7

C. 9

D. 11

1.15 第 15 题

一棵 2025 个结点的无根无权树中，所有**无序**点对的距离和的最大值除以 7 的余数是 ▲ 。

A. 1

B. 2

C. 3

D. 4

2 阅读程序（无特殊说明时判断 1.5 分，选择 3 分，3 题共 40 分）

2.1 第 1 题（12 分）

```
#include<bits/stdc++.h>
using namespace std;
int a[60], k, c[10];
int main(){
    scanf("%d", &k);
    for(int i = 0 ; i < k ; i++) scanf("%1d", &a[i]);
    for(int i = 0 ; i < k ; i++){
        if(a[i] == 1 || (a[i] != 2 && a[i] % 2 == 0) || a[i] == 9){
            printf("%d\n", a[i]);
            return 0;
        }
    }
    for(int i = 1 ; i < k ; i++)
        if(a[i] == 2 || a[i] == 5){
            printf("%d%d\n", a[i - 1], a[i]);
            return 0;
        }
    for(int i = 0 ; i < k ; i++){
        if(c[a[i]]){
            printf("%d%d\n", a[i], a[i]);
            return 0;
        }
        c[a[i]] = 1;
    }
```

```
25      for(int i = 1 ; i < k ; i++)
26          if((a[0] == 2 || a[0] == 5) && a[i] == 7){
27              printf("%d%d\n", a[0], a[i]);
28              return 0;
29          }
30      return 0;
31  }
```

输入数据满足 $1 \leq k \leq 50$。

2.1.1 判断题

1.（1 分）该程序的功能是将一个数依据某一规则转化为某个合数。

2. 输入3 675时，输出为 65。

3. 输入4 2048时，输出为 0。

2.1.2 选择题

4.（2 分）输入12 379379379379时，输出为 ___▲___。

A. 33

B. 99

C. 9

D. 77

5. 输入40 2373737373737373737373737373737373737373时，输出为 ___▲___。

A. 27

B. 23

C. 77

D. 33

6. 某两次输出结果的和为 26，且第一次输出为一位数，则第一次输出可能为 ___▲___。

A. 0

B. 1

C. 6

D. 8

2.2 第 2 题（13 分）

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

int main() {
    int n;
    cin >> n;
    vector<int> a(n);

    for (int i = 0; i < n; i++) {
        cin >> a[i];
    }

    int x = 0, y = 0;

    for (int i = 0; i < n; i++) {
        if (i % 2 == 0) {
            x = x + a[i];
            y = y - a[i];
        } else {
            x = x - a[i];
            y = y + a[i];
        }
    }

    auto f = [&](int p, int q) {
        if (p > q)
            return p - q;
        else
            return q - p;
    };

    int ans = 0;

    for (int i = 0; i < n; i++) {
        if (i % 2 == 0) {
            ans += f(x, a[i]);
        } else {
            ans += f(y, a[i]);
        }
    }
```

```
42
43      cout << ans << endl;
44      return 0;
45  }
```

2.2.1 判断题

1. 若输入数组a中所有元素均为 0，则程序输出的ans为 0。

2. 若将第 14–22 行的循环改为从i=1开始到i<=n，程序输出结果不改变。

2.2.2 选择题

3. 当输入为 4\n 2025 2025 -2025 -2025 时，输出为 __▲__。

A. 0

B. 2025

C. 4050

D. 8100

4. 当输入为 5\n 1 2 3 4 5 时，输出为 __▲__。

A. 4

B. 10

C. 16

D. 22

5. （4 分）若将第 16 行改为x = x + 2 * a[i];，输入 3\n 3 1 2 时，输出为 __▲__。

A. 8

B. 36

C. 14

D. 18

2.3 第 3 题（15 分）

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  int n, cnt[12];
4  int a, b, c, d;
5  int main(){
6      scanf("%d%d%d%d%d", &n, &a, &b, &c, &d);
7      for(int i = 1 ; i <= n ; i++){
```

```
        int len;
        scanf("%d", &len);
        cnt[len]++;
    }
    if(cnt[5] != c || cnt[7] != d){
        puts("0");
        return 0;
    }
    int num_3 = cnt[3] + 2 * cnt[9];
    int min_2, max_2, min_3, max_3;
    min_3 = num_3 + cnt[6];
    max_3 = num_3 + 2 * cnt[6];
    if(b < min_3 || b > max_3){
        puts("0");
        return 0;
    }
    int trans_6 = b - min_3;
    min_2 = cnt[6] - trans_6;
    max_2 = min_2 + cnt[2] + 2 * cnt[4] + 4 * cnt[8];
    if(a < min_2 || a > max_2){
        puts("0");
        return 0;
    }
    puts("1");
    return 0;
}
```

输入数据满足 $1 \le n \le 10^5$，$1 \le \text{len} \le 9$，$0 \le a, b, c, d \le 4 \times 10^5$，且所有输入均为正整数。

2.3.1 判断题

1.（1 分）本程序的时间复杂度为 O(1)。

2. 输入`5 3 3 0 0\n2 6 6 6 8`时，运行第 27 行后，`max_2 - min_2 = 5`。

2.3.2 选择题

3.（2.5 分）若 $a = b = c = d = 10$，输出为 1，则 n 的最小值为 ▲。

A. 27

B. 28

C. 29

D. 30

4. 实际上，当 $\lambda * n < a + b + c + d$（$\lambda$ 是参数）时，即可直接输出 0，λ 的最小值为 ▲。

A. 2

B. 3

C. 4

D. 5

5. 若输入的 len = [2, 4, 6, 6, 6, 6, 8, 8]，c = 0，d = 0，且 a，b 均在 [0, 9] 内随机均匀取值，则输出为 1 的概率为 __▲__。

A. 0.37

B. 0.38

C. 0.39

D. 0.40

6.（4 分）若输入的 n = 9，b = 6，c = 0，d = 0，且所有 len 的总和为 40，并满足 len 均在`{2, 4, 6}`内取值，假定输入的 len 单调不降，则所有可能使输出为 1 的输入种数为 __▲__。

A. 25

B. 26

C. 27

D. 28

3 完善程序（共两道题 30 分）

3.1 反色编码（15 分）

为了形式化地描述颜色，我们引入颜色值，用三元组 $(r, g, b)(0 \le r, g, b \le 255)$ 表示一种颜色，其中 r, g, b 分别为该颜色的 R/G/B 值，皆为十进制整数。对于该颜色，定义其反色的颜色值为 $255 - r, 255 - g, 255 - b$。

另一方面，十六进制颜色码是一个长度为 7 的字符串。具体而言：

- 字符串的第一位是 `#`，为颜色码标识符。
- 字符串的第二、三位组成的十六进制数等于十进制下该颜色的 R 值。
- 字符串的第四、五位组成的十六进制数等于十进制下该颜色的 G 值。
- 字符串的第六、七位组成的十六进制数等于十进制下该颜色的 B 值。

现在你需要把一组十六进制颜色码转化为其反色的十六进制颜色码。

试补全程序。

```
#include <cstdio>
using namespace std;
char c[6];
/* Blank 1 */ to_t_o(char c1, char c2){
    int a, b;
    if(c1 >= 'A') a = 10 + (c1 - 'A');
    else a = c1 - '0';
    if(c2 >= 'A') b = 10 + (c2 - 'A');
    else b = c2 - '0';
    /* Blank 2 */;
}
void to_t_t(int n){
    int a = n / 16;
    if(a >= 10) printf("%c", a - 10 + 'A');
    else printf("%d", a);
    a = /* Blank 3 */;
    if(a >= 10) printf("%c", a - 10 + 'A');
    else printf("%d", a);
}
int main(){
    scanf("#");
    for(int i = 0 ; i < 6 ; i++)
        scanf("%c", &c[i]);
    printf("#");
    /* Blank 4 */ {
        if(i % 2) /* Blank 5 */;
    }
    return 0;
}
```

1./* Blank 1 */ 应填 __▲__。

A. `int`

B. `void`

C. `char`

D. `inline`

2./* Blank 2 */ 应填 __▲__。

A. `return a + b * 16`

B. `return a + b % 16`

C. `return a * 16 + b`

D. `return`

3./* Blank 3 */ 应填 ▲。

A. `n % 16`

B. `n - 16 * a + 1`

C. `(n + 1) % 16`

D. `(n + 1) / 16`

4./* Blank 4 */ 应填 ▲。

A. `for(int i = 1 ; i <= 7 ; i++)`

B. `for(int i = 0 ; i < 6 ; i++)`

C. `for(int j = 0 ; j < 6 ; j++)`

D. `for(int i = 0 ; i < 6 ; i += 2)`

5./* Blank 5 */ 应填 ▲。

A. `to_t_o(255 - to_t_o(c[i-1], c[i]))`

B. `to_t_o(255 - to_t_t(c[i-1], c[i]))`

C. `to_t_t(255 - to_t_t(c[i-1], c[i]))`

D. `to_t_t(255 - to_t_o(c[i-1], c[i]))`

3.2 相通变换（15 分）

给定 n 个数 $a_1, a_2, \ldots, a_n$。单次操作内，你可以选择 $i \neq j$，并使得 $a_i \leftarrow a_i - 1$，同时 $a_j \leftarrow a_j + 1$。求最小操作次数，使得存在整数 $x > 1$，满足这 n 个数均能被 x 整除。

注意：任意时刻，你均需要保证 $a_i \geq 0$。提示：0 能被任何正整数整除；可以证明，一定有解。

例如，当 $n = 5, a = [1, 2, 3, 4, 5]$ 时，最小操作次数为 2，具体解释为：

- 初始 $a = [1, 2, 3, 4, 5]$；
- 第一次，选择 $i = 1, j = 2$，$a = [0, 3, 3, 4, 5]$；
- 第二次，选择 $i = 4, j = 5$，$a = [0, 3, 3, 3, 6]$；
- 此时，取 $x = 3$，符合要求。可以证明，2 是最小的操作次数。

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <cmath>
#include <climits>
using namespace std;
typedef long long LL;
```

```
8
9  vector<LL> getPrimeFactors(LL n) {
10     vector<LL> factors;
11     while (n % 2 == 0) {
12         if (factors.empty() || factors.back() != 2)
13             factors.push_back(2);
14         n /= 2;
15     }
16     for (LL i = 3; /* blank 1 */; i += 2) {
17         while (n % i == 0) {
18             if (factors.empty() || factors.back() != i)
19                 factors.push_back(i);
20             n /= i;
21         }
22     }
23     if (n > 2) factors.push_back(n);
24     return factors;
25 }
26
27 int main() {
28     ios::sync_with_stdio(false);
29     cin.tie(0);
30
31     int n;
32     cin >> n;
33     vector<LL> a(n);
34     LL sum = 0, max_val = 0;
35     for (int i = 0; i < n; i++) {
36         cin >> a[i];
37         sum += a[i];
38         max_val = max(max_val, a[i]);
39     }
40     vector<LL> primes = getPrimeFactors(sum);
41     LL ans = LLONG_MAX;
42     if (primes.empty()) {
43         ans = 0;
44     } else if (primes.size() == 1 && primes[0] == sum) {
45         ans = /* blank 2 */;
46     }
47     int ln = primes.size();
48     for (int j = 0 ; j < ln ; j++) {
49         int x = primes[j];
```

```
        if (x == 1) continue;
        vector<LL> rem;
        LL need_rem = 0;
        for (int i = 0; i < n; i++) {
            LL r = /* blank 3 */;
            if (r > 0) {
                rem.push_back(r);
                need_rem += r;
            }
        }
        sort(rem.begin(), rem.end());
        LL cur = 0, l = 0, r = rem.size() - 1;
        while (l < r) {
            LL take = /* blank 4 */;
            rem[l] -= take;
            rem[r] += take;
            cur += take;
            /* blank 5 */
            r -= (rem[r] >= x);
        }
        ans = min(ans, cur);
    }
    cout << ans << endl;
    return 0;
}
```

3.2.1 选择题

1. /* blank 1 */ 应填 __▲__。

A. `i < n / i`

B. `i * i < n`

C. `i * i <= n`

D. `i < sqrt(n)`

2. /* blank 2 */ 应填 __▲__。

A. `sum - max_val`

B. `sum / max_val`

C. `max_val`

D. `sum`

3. /* blank 3 */ 应填 __▲__。

A. `x % a[i]`

B. `a[i] - x`

C. `a[i] / x`

D. `a[i] % x`

4. /* blank 4 */ 应填 __▲__。

A. `min(rem[l], rem[r])`

B. `min(rem[l], x - rem[r])`

C. `max(rem[l], rem[r])`

D. `min(x, rem[r])`

5. /* blank 5 */ 应填 __▲__。

A. `l += (rem[l] == 0)`

B. `while(!rem[l]) l++`

C. `l -= (rem[l] > 0)`

D. `while(rem[l]) l--`