CSP 第二轮考试注意事项及防爆零指南

CSP 第二轮考试主要考察学生的信息学知识应用、问题建模和上机编程能力。由于学生需要进行真实的代码编写，可能会遇到各种问题，其中包括一些较为细致的非知识和非能力性问题。如果没有按照要求处理，即使能够完成所有题目，最终成绩也将是“零分”。

一 、程序题目提交要求

拿到试卷后，一定要认真仔细的阅读试卷注意事项 ：

2022 CCF 非专业级软件能力认证

CSP-J/S 2022 第二轮认证

入门级

时间： 2022 年 10 月 29 日 08:30 ～ 12:00

题目名称	乘方	解密	逻辑表达式	上升点列
题目类型	传统型	传统型	传统型	传统型
目录	pow	decode	expr	point
可执行文件名	pow	decode	expr	point
输入文件名	pow.in	decode.in	expr.in	point.in
输出文件名	pow.out	decode.out	expr.out	point.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 MiB	512 MiB	512 MiB	512 MiB
测试点数目	10	10	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	pow.cpp	decode.cpp	expr.cpp	point.cpp

编译选项

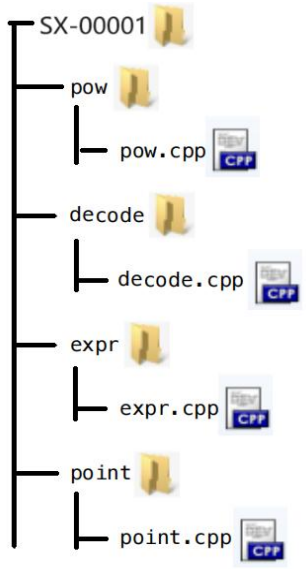
对于 C++ 语言	-O2 -std=c++14

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考各省的具体要求。
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. 选手提交的程序源文件必须不大于 100KB。
7. 程序可使用的栈空间内存限制与题目的内存限制一致。
8. 全国统一评测时采用的机器配置为：Inter(R) Core(TM) i7-8700K CPU @3.70GHz，内存 32GB。上述时限以此配置为准。
9. 只提供 Linux 格式附加样例文件。
10. 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此为准。

CSP 第二轮考试采用 NOI 赛制，即选手在考试过程中可以根据题目写出对应的程序，但是考试过程中没有办法测评，这样就不知道每份代码是否正确。而是在考试结束后，选手需要严格按照要求提交一份包含每个题目程序文件的文件夹，文件夹目录错误或者文件夹名错误都会导致 0 分。

例如四道题名称分别为 pow、decode、expr、point，则提交代码时，文件目录如下：



其中：

SX-00001 为选手文件夹，一般以**选手的考号命名**，具体参照考场要求；

pow 和其余三个文件夹为每个题目的文件夹，一般以**题目的英文名称**命名；

pow.cpp 等三个 cpp 文件为源代码，一般的命名规则为：**题目英文名称.cpp**；

一定要注意：

1、听从监考老师要求，**根据老师的要求存放文件**。提交代码时，要检查**文件夹目录结构是否正确，文件夹名字是否写错**（严格区分大小写），**源代码(cpp)的名称是否写错**。（一般情况下以"SX-考号"为名字建立一个文件夹，此文件夹下分别建立四道题目的文件夹，这四个文件夹内各自存放一个题目的源程序代码。）

2、有些 Windows 系统的电脑的扩展名是隐藏的，一定要先操作电脑设置，**让扩展名显示出来**。比如有些机器隐藏了扩展名，选手将程序命名为 abc.cpp，实际上是 abc.cpp.cpp；选手将输入文件命名为 abc.in，实际上是 abc.in.txt。

3、文件夹内不能有其余文件；

4、考试结束后一定要寻找、以及要求监考老师确认自己的文件夹是否提交成功，一定不要不好意思要求监考老师帮你检查。

二、程序中可能存在爆 0 的非能力问题

1、头文件问题

CSP考试的编译环境支持万能头文件<bits/stdc++.h> ，且不会因编译时间过长导致零分（判题系统测试的是程序运行时间而不是编译时间），所以老师还是推荐大家在考场上直接使用万能头文件。

万能头文件为 <bits/stdc++.h>，在 Windows 环境中 <bits\stdc++.h> 也可以通过编译，但是在判题时所使用 Linux 环境中会编译失败导致0分。因此在使用万能头文件的时候一定要注意斜杠的方向（**正确的斜杠为“除号”对应的斜杠，也即向右倾倒的斜杠，“/”为正确**）。

2、文件输入输出

(1) 在 CSP 第二轮考试中，**要求解题程序使用文件输入输出，**可以在试题的第一页查看每道题目要求的「输入文件名」和 「输出文件名」。在 C++ 中可以通过使用 freopen 函数实现输入输出的重定向操作，**freopen要写在 main 函数内的第一行**。

例如第一题 「乘方」 可以写出程序框架：

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

int main() {
    freopen("pow.in", "r", stdin); // 重定向输入，从  pow.in 文件中读入数据
    freopen("pow.out", "w", stdout); // 重定向输出，将  cout 的结果写入  pow.out 文件内

    // code

    return 0;
}
```

(2) freopen 需要的头文件为<bits/stdc++.h>

freopen 代码格式为：

freopen("xxx.in", "r", stdin);

freopen("xxx.out", "w", stdout);

其中 xxx 为题目的英文名，例如题目名字为 pow，则应写为：

freopen("pow.in", "r", stdin);

freopen("pow.out", "w", stdout);

注意，这里的函数名称和参数不能写错，任意一个地方出错都会导致程序爆 0。

比如，abc.in 不能写成 abc.txt，也不能写成 abc.in.txt。

文件名"xxx.in"，"xxx.out"都带有英文双引号，参数“r”，“w”都带有英文双引号，参数 stedin、stdout 都不带引号。

“r”不能写成“read”，也不能写成单引号’r’，注意一定是双引号的“r”和“w”！

两行 freopen 中的三个参数，是分别相反的，"xxx.in"文件对应着"xxx.out"文件，读取"r"对应着写入"w"，标准输入 stdin 对应着标准输出 stdout。

（3）如果你需要测试样例的时候 ，需要将给出的样例输入文件pow1.in 放在和pow.cpp 同文件夹下（记得备份）， 然后程序内修改为freopen("pow1.in", "r", stdin); freopen("pow1.out", "w", stdout); ，这样程序就可以从 pow1.in 读入数据 ，程序正常运行后会在同目录下自动生成 pow1.out 文件 ，查看输出结果。

注意：如果进行上述操作，在提交代码的时候，一定要检查使用的文件名是否改回为正确，例如：pow.in 、pow.out 正确 ，而使用 pow1.in、pow1.out 就会导致零分。

（4）如果在测试的时候，不想使用文件输入输出，而只使用日常的键盘、屏幕作为输入输出方式，可以将这两行 freopen 代码注释掉，但是在提交之前一定要将注释撤销，然后再编译运行一下代码，确保程序正确。**忘记撤销对 freopen 的注释会导致代码为 0 分，最终提交的程序里面一定要有正确的 freopen()做文件输入输出重定向。**

3、系统变量名冲突

平常写程序的时候一般都会加上标准的命名空间 using namespace std; ，这个时候就不能在全局变量上定义 **y0, y1, yn, j0, j1, jn, next, time** 等变量，因为它们已经在标准库中被定义过了，用这些变量名会导致编译错误，程序得分 0 分。

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

int y0, y1; // 不能这样定义，会报编译错误
int next[1010]; // 不能这样定义，会报编译错误

int main() {
    freopen("pow.in", "r", stdin); // 重定向输入，从 pow.in 文件中读入数据
    freopen("pow.out", "w", stdout); // 重定向输出，将 cout 的结果写入 pow.out 文件内

    // code
```

```
    return 0;
}
```

注意：

（1）在 Windows 环境下不会报错 ，在 Linux 环境下会报错，**判题环境为Linux所以会导致0分**。
（2）可以定义在局部变量，不过也不推荐。

4、忘记删除过程调试代码

在平时解决难题的过程中，我们经常采用输出调试法，也就是输出程序运行过程中一些重要步骤的运算结果，以便找出程序出错的地方。然而，这样做会导致程序中存在大量不必要的输出语句，因此在最终提交程序时，一定要记得删除这些调试代码（即提交代码时要记得删除自己添加的测试语句）。

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

int main() {
    freopen("pow.in", "r", stdin);
    freopen("pow.out", "w", stdout);
    int n, sum;
    cin >> n;
    for (int i = 1; i <= n; i++) {
        int x;
        cin >> x;
        sum += x;
        cout << i << " : " << sum << endl; // 调试代码，提交前记得删除
    }
    cout << sum << endl;
    return 0;
}
```

例如输入：
3
1 2 3

程序会输出：
1 : 1
2 : 3
3 : 6
6

注意，上述代码是统计 n 个数的和， 实际错误之处是因为 sum 没有赋初值 0，这一点在考试中也需要注意（定义局部变量时未初始化，局部变量的值为随机数，需要手动给局部变量赋初值）。在考试的过程中，也需要根据数据范围和数据类型，判断变量 sum 应使用 int 、long long 还时 double 类型。

三、未认真读题造成的粗心错误

1、未按照题目指定规则输出

错误场景1：

看清楚题目要求是以空格隔开还是以换行隔开，按要求输出。

错误示例：

题目要求输出 1 行，每个数据之间以空格隔开，正确的数据应该为：

1 2 3

部分粗心的选手可能写成以下情况，导致零分：

1
2
3

错误场景 2：

有些题目要求无解输出-1，不要忘记无解情况对应的输出。

错误场景 3：

注意输出时的大小写，题目要求正确输出 Yes；错误输出 No，不要粗心写成 YES 和 NO。

2、未按照题目定规则输入

错误场景 1：

部分题目存在**多组测试数据**，对于多组数据的题目要注意以下几点：

不要错看成一组数据，导致没有用循环输出，代码爆零。

处理每组数据前，**需要初始化的变量要记得初始化，例如累加器，计数器等；需要清空的数组要记得清空。**

例：输入 n 组数据，每组数据输入两个正整数，请输出每组数据中两个正整数的和。

正确写法

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

int main() {
    freopen("pow.in", "r", stdin);
    freopen("pow.out", "w", stdout);
    int n, sum;
    cin >> n;
    while (n--) {
        int x, y;
```

```
        cin >> x >> y;
        cout << x + y << endl;
    }
    return 0;
}
```

错误写法

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

int main() {
    freopen("pow.in", "r", stdin);
    freopen("pow.out", "w", stdout);
    int x, y;
    cin >> x >> y;
    cout << x + y << endl;
    return 0;
}
```

错误场景 2：

粗心未看清 「输入格式」 中对数据含义的描述，导致输入错误。

例：程序需要输入一个 n 行 m 列的二维数组。

输入格式：第一行输入两个正整数 **m, n 分别表示二维数组的列和行**……

错误代码

```
int n, m;
cin >> n >> m; // 输入格式中第一个数是二维数组的「列」，第二个数是二维数组的「行」
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= m; j++) {
        cin >> a[i][j];
    }
}
```

3、数据范围看错

要看清楚数据范围，**数组开太小容易导致数组越界，开太大容易导致空间超限导致 0 分**，一般比题目要求多开 5 个空间即可。

例：题目要求输入 100000 个数据 ，并存储在数组中， 因为 0 比较多 ，可能错写为：

```
int num[10005]; // 少一个 0
```

```
int num[1000005]; // 多一个 0
```

推荐写法：

```
const int N = 1e5 + 5;    // “1e5”代表 10 的 5 次方，“+5”代表多 5 个。
```

```
int num[N];
```

4、数据类型使用错误

要预估题目中数据的范围，int 类型的变量能存储的最大值为 21 亿（约为 2×10^9）左右，运算过程中有可能超过这个数值需要用 long long 类型或者高精度算法。

例：题目要求你计算两个正整数的乘积， 已知数据范围 $1 \leqslant x, y \leqslant 10^6$ ，你可能会错写为：

```
int x,y;
cin >> x >> y;
cout<<x*y<<endl;
```

在这段代码中 ，x, y 如果只存储数值 ，x 和 y 的值都不超过整型可存储的最大值，那么程序没有问题。但是要进行乘法运算，计算的结果就可能溢出（ 超出了 int 类型可以存储的最大值 2147483647） 。例如考虑极端情况 $x = y = 10^6$ ，那么 $10^6 \times 10^6 = 10^{12}$ 超过 int 的范围 ，导致计算结果错误。

一般在考试中，不是所有数据都是极端数据 ，所以发生这类错误 ，还是可能得到部分分数的，但也要尽量去避免数据溢出导致的丢分。

正确写法 1：

```
long long x,y;
cin >> x >> y;
cout<<x*y<<endl;
```

正确写法 2：

```
int x,y;
long long z;
cin >> x >> y;
 z=1LL*x*y;
cout<<z<<endl;
```

正确写法 3：

```
int x,y;
cin >> x >> y;
```

```
cout<< 1LL*x*y<<endl;
```

“1LL”的意思是“常量数字 1 为 long long 类型”，“1LL*x*y”代表按照计算顺序，longlong 类型的常量 1（即 1LL）先与 x 相乘，因为精度高的数据和精度低的数据进行运算会自动将中间结果转换为精度高的类型暂存起来，这个 1LL*x 的结果即为 longlong 类型，然后 1LL*x 的 longlong 类型中间结果再与 y 相乘，结果也为 longlong 类型，再进行输出。

四、程序性能问题

1、题目存在大量的输入/输出数据

在 C++ 中，cin/cout 对数据的读写速度远低于 scanf/printf ，**如果题目中需要输入/输出的数据超过 10^5 个时，使用 cin/cout 可能导致程序超时。**

可以使用如下两种办法解决：

<1>选择使用 scanf/printf

（**未学过 scanf 的同学不要使用这个语法，后续学习并熟练掌握后再使用**）

使用 scanf 进行输入，读入数据的速度比较快。但一定要记清楚不同数据类型对应的格式控制符。

例：输入一个整数，存放到整型变量 a 中：scanf("%d",&a); 一定要记得加取地址符！

输出整型变量 a 的值： printf("%d",a);

常用格式控制符如下图：

数据类型	格式控制
int	%d
long long	%lld
double	%lf
long double	%llf
K位小数 float	%.klf

<2>手动实现快速读入和快速输出函数

快速读入模板：

```
inline void read(int &x){
    char ch;
    bool flag=false;
    for (ch=getchar();!isdigit(ch);ch=getchar())if (ch=='-') flag=true;
    for (x=0;isdigit(ch);x=x*10+ch-'0',ch=getchar());
```

```
    x=flag?-x:x;
}
```

快速输出模板：

```
inline void write(int x){
    static const int maxlen=100;
    static char s[maxlen];
        if (x<0) {   putchar('-'); x=-x;}
    if(!x){ putchar('0'); return; }
    int len=0; for(;x;x/=10) s[len++]=x % 10+'0';
    for(int i=len-1;i>=0;--i) putchar(s[i]);
}
```

如果题目中需要输入/输出的数据超过 10^5 个时，学生可从上述两种方法中选择自己学过的且掌握熟练的方法解决输入输出超时问题。

2、程序的复杂度过高

考试时每个题目都会给出**时间和空间限制**，选手要保证自己写的程序在限制内可以正确执行。

2022 CCF 非专业级软件能力认证

CSP-J/S 2022 第二轮认证

入门级

时间：2022 年 10 月 29 日 08:30 ～ 12:00

题目名称	乘方	解密	逻辑表达式	上升点列
题目类型	传统型	传统型	传统型	传统型
目录	pow	decode	expr	point
可执行文件名	pow	decode	expr	point
输入文件名	pow.in	decode.in	expr.in	point.in
输出文件名	pow.out	decode.out	expr.out	point.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 MiB	512 MiB	512 MiB	512 MiB
测试点数目	10	10	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	pow.cpp	decode.cpp	expr.cpp	point.cpp

编译选项

对于 C++ 语言	-O2 -std=c++14

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考各省的具体要求。
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. 选手提交的程序源文件必须不大于 100KB。
7. 程序可使用的栈空间内存限制与题目的内存限制一致。
8. 全国统一评测时采用的机器配置为：Inter(R) Core(TM) i7-8700K CPU @3.70GHz，内存 32GB。上述时限以此配置为准。
9. 只提供 Linux 格式附加样例文件。
10. 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此为准。

（1）时间复杂度

一般评测机在 1 秒内大概可以处理 5 亿次简单运算，保险起见选手要确保程序的简单运算次数不要超过 1 亿次，即 10 的 8 次方。

下图为 1 秒的运行时限内，算法的时间复杂度与对应的数据规模表：

算法时间复杂度	允许的数据范围
$O(\log n)$	∞
$O(n)$	$\leq 10^8$
$O(n \log n)$	$\leq 10^6$
$O(n\sqrt{n})$	$\leq 10^5$
$O(n^2)$	$\leq 10^4$
$O(n^3)$	≤ 300
$O(2^n)$	≤ 25
$O(3^n)$	≤ 15
$O(n!)$	≤ 11

根据数据范围可以大致判断应选用怎样的算法才能不超时；

根据自己代码的时间复杂度和数据范围可估算出自己的代码是否会超时。

（2）空间复杂度

常见变量类型所占内存大小：

char、bool 占 1 字节；int 占 4 字节；long long、double 占 8 字节；

常用内存单位换算：1MB=1024KB 1KB=1024B 1Byte=8bit

通常题目会限制内存不能超过 256M 或 512M。要会估计内存。比如全局数组 int a[1000000] 占内存 1000000 * 4 / (1024 * 1024) M ≈ 4M。

如果一个题目空间限制为 256 MB ，那么它最多可以使用的字节数：256 × 1024 × 1024 = 268435456 字节 ，最多可以声明长度为 268435456 ÷ 4 = 67108864 的 int 类型数组。

注意：**内存哪怕超一个字节也会直接零分，**并且除了自己开的变量以外还要预留一定空间，所以一定不要乱开大容量的数组。

五、其他易犯非能力错误

（1）**声明局部变量要记得初始化,**该局部变量的值是一个随机数,最好使用全局变量或给局部变量赋初始值。

（2）**数组建议用全局数组**，千万不要放到 main 函数里面，二维数组很容易就爆了，全局数组可用的内存空间比局部数组可用的内存空间大很多很多。

（3）**浮点类型一律使用 double,不要使用 float**，经常会由于 float 精度不够导致丢分。

（4）输出的时候如果用 printf ，千万千万注意 long long 类型的变量一定是 printf("%lld", xxx)，千万不要写成了 %d 。

（5）**注意定义数据的类型**。小心 double 定义成 int，int 定义成 char 等。

（6）**运算符优先级**一定要注意，不确定优先级时多用小括号去控制运算先后顺序。

（7）涉及取模的问题，一定要在**运算过程中取模**，数据爆了再取模就是 0 分。

（8）读入**带空格字符串**时，用字符数组+fgets 输入方法，或用字符串+getline 输入方法，可以将字符串内的空格读入。

（9）**严格禁止使用 gets 函数**，会导致爆 0。

（10）要看清字符数组的**下标是从 1 开始使用，还是从 0 开始使用**。

六、考试结束前一定要做的

在距离考试结束还有最后 10 分钟时，就不要再改动程序了，请检查如下几项：

1、文件存放目录是否正确：以考号命名的文件夹内有四个题目文件夹，每个题目文件夹内有一个 cpp 文件，以及文件夹和文件名称是否按符合题目要求。

2、头文件是否正确完整。

3、是否正确的书写了文件输入输出语句，以及对应的文件名是否正确；

4、是否注删掉了中间调试用的代码；

5、对每个题目的程序再编译、运行一下，至少确保程序不要出现编译错误问题；

6、提交的文件夹内是否还有不需要的文件。

以上为 CSP 第二轮考试注意事项及防爆零指南，要求大家：

1、打印出来，要在微信群中轻声读一遍，并在读的过程中大脑积极思考，大脑中映射敲代码时的操作过程，加深理解。

2、不理解的部分，需要上机实践操作的步骤一定要上机实践。