3rd Annual Cornell University
High School Programming Contest Rules

Team formation:
To compete you should be a team of three students enrolled in the same institution. You may bring with you as much printed material as you like (books, old codes you have developed, course notes, etc.), but no electronic device will be allowed. That means that any laptop, phone, smartwatch, iphone, USB drive or any other form of media will not be allowed on the competition floor.

The problem set:
The problem set will vary in difficulty. **The problems are NOT presented in order of difficulty.** You may solve your problems in any order you want, and each problem awards you with the same amount of points. Some problems will give partial credit for teams that solve it only partially. More about grading later.

Submitting:
After reading a problem statement, if you decide to approach it, you should code your solution in one of the accepted languages (C, C++, Java, Python, Ruby, Go, and several other). After you test your solution in your computer, and think that it is correct, you may submit your code for judging. Upon receiving your code the automatic system will:

1. Compile your code (if not in Python)
2. Execute your code with a prepared set of hard input cases (that are hidden from the contestants)
3. Compare the output your program generated with the expected output for each case.

Note that an automatic judge will do this; no human will read your code. After the judging (usually a few seconds, but may be more depending on the judging queue) you will receive a verdict for each input case, which can be:

- **Compilation Error** - Your code didn't compile
- **Time Limit Exceeded** - Your code took longer to run than the allowed time limit
- **Wrong Answer** - Your code ran in time, but it output a wrong solution for at least one of the test cases.
- **Runtime Error** - Your code ran into a runtime error, for instance, a division by 0, or a syntax error in the case of Python.
- **Correct** - Your code compiled and ran correctly, and output a correct answer.

We will be using HackerRank as our judging system. To access this particular contest you will need an account on HackerRank, and go to the following address:

https://www.hackerrank.com/contests/cornell-university-high-school-programming-contest/challenges to sign up. Once it's time you will be able to see the challenges.

Scoring:
Each problem is worth up to 100 points. Some problems will appear more than once in the list of problems in the form: "Small <Problem Name>", "Medium <Problem Name>" and "Large <Problem Name>". In the case this happens, a code that correctly solves the "Large" instance will also solve all the other ones. This is a way to give partial credit.

In the case a problem has only one version, it will give you 100 points. If it has a small and a large version it will give you 40 and 60 points respectively, and in the case you have small/medium/large versions you will receive 30/30/40. To receive the points you need to output the correct answer to **ALL** the hidden input cases. A single "Time Limit Exceeded" or "Wrong Answer" is enough to take all your points away. Upon receiving a negative verdict you have the option to fix your code and resubmit.

The first criterion for scoring is the total number of points. If team A has 340 points and team B has 300 points, team A will be ahead of team B in the ranking, regardless of which problems they solved, and of the time it took them.

The tie breaking is done by considering the "penalty". Penalty has two components:

1. You will receive a penalty of 20 for each submission that is not accepted for **a problem or version of a problem you eventually solve.** If you don't eventually solve it, you won't be penalized for wrong submissions.
2. If you receive a "YES" on a problem **X** minutes after the competition started, you will receive a penalty of **X**.

In case of ties in number of problems, teams will be ranked by smallest penalties. In the very unlikely case of ties in both number or problems and penalties, the judges may decide to make a (subjective) determination based on the quality of the submitted code.

Clarifications: In case you have a question about a problem statement, you may raise your hand and a volunteer will answer your question. There are **NO PENALTIES** for asking for clarifications, and you can ask for as many as you want. Most of the times the answer will be "Read the statement carefully", but do ask everything you want in case of any doubt.

At your station you will have at your disposal:

- One computer (note that there will be only one computer per team of three people)
- Three copies of the problem set, printed. These copies are yours to keep, and you can write on them as you will.
- Sheets of paper and pens.

Apart from that, you will be able to print anything you want (for instance, to analyze code while another team member is working on another problem on the computer). You are ***not*** allowed to use the Internet, however. Communication of any sort with another team, an acquaintance or a complete stranger online will result in immediate disqualification.

Some Quick Hints

- For editing, you have many choices, including
    - gedit: a simple to use graphical editor. Can be started from a terminal window.
    - Eclipse: a complete programming environment. Should be in sidebar.
    - Vim or emacs: Really powerful, but old school editors. Don't use them if you never did before.

  You can ask a staff member for help with these editors.
- If you need to go to the bathroom, please ask one of the attendants to accompany you.
- Don't hesitate before asking for any clarification on a detail of a problem statement you don't understand.
- All programs should read from standard input and write to standard output.
- There is a running time limit of 5 seconds on each solution, which should be plenty.
- Until 30 minutes before the end of the contest (or until we run out of helium or balloons), you will get a balloon for each solved problem, in a color that corresponds to the problem you solved. The colors of the problems are not disclosed, so unless you have a balloon of the same color you cannot tell which problems other teams have solved.
- The progress of other teams (in terms of score) can also be seen on the scoreboards until 30 minutes before the end.

Each solution takes a single input (read from standard input), and produces a single output (written to standard output).

Spaces, new-lines, etc. are generally ignored except to delimit the input fields. Read the instructions carefully to understand the format of the input and output. We provide for each problem at least one sample input and output.

Reading From the Standard Input:

Suppose the input starts with a number N, followed by N integers, and you should output the sum of those integers. This is an over-simplified problem to give you an example of accepted solution.

For example, if the input is:
3
1
2
5

The output should be:
8

In Java:

```java
import java.util.*;

class ClassName { // use any ClassName

    public static void main(String [] args) {

        Scanner sc = new Scanner(System.in); // prepare to read from standard input

        int N = sc.nextInt();

        int ans = 0;

        for (int i = 0; i < N; i++) {

            ans += sc.nextInt();

        }

        System.out.println(ans);

    }

}
```

In C++:

```cpp
#include <iostream>

using namespace std;


int main() {

    int N;

    cin >> N;

    int ans = 0;

    for (int i = 0; i < N; i++) {

        int x;

        cin >> x;

        ans += x;

    }

    cout << ans << endl;
```

```
    return 0;
}
```

In C:

```
#include <stdio.h>

int main() {
    int N;
    scanf("%d", &N);
    int ans = 0;
    for (int i = 0; i < N; i++) {
        int x;
        scanf("%d", &x);
        ans += x;
    }
    printf("%d\n", ans);
    return 0;
}
```

In Python:

```
N = int(input())
ans = 0
for i in range(N):
    ans += int(input())
print(ans)
```