Judging Director: Abu Saleh
Associate Judging Director: Masum Bin Hossain
Slot A Judging Director: Md. Albin Hossain
Slot A Associate Judging Director: Sabbir Ahamed Mridha

Problem A: LET'S START

Problem Setter: Sabbir Ahamed Mridha
Reviewer: Samia Haque Tisha
Alternate Writer: Md Shefat Al Mahmud

Problem Analysis:

- Observation: All you have to do is write a code which will print “Ada Lovelace” (without quotes).
- Solution Idea: Read the problem statement and just copy the code from the statement. Just change the printed string from “Programming is the best!!” to “Ada Lovelace”.

Special Thanks: Samia Haque Tisha

Source Code:

```c
#include<stdio.h>

int main() {
    printf("Ada Lovelace");

    return 0;
}
```

Time complexity: O(1)

Problem B: Trailing Zero Offer

Problem Setter: Md Shefat Al Mahmud
Reviewer: Samia Haque Tisha
Alternate Writer: Sourav Halder

Problem Analysis:

- Observation: The numbers that have trailing zeros are multiples of 10. The numbers between 1 to 99 having trailing zeros are 10,20,30,40,50,60,70,80, and 90 and the numbers are multiples of 10.
- Solution Idea: If a number is divided by 10, that number is a multiple of 10. In this program, an if-else statement needs to be used to check whether n mod 10==0 is true or not. If the expression is true, the number is a multiple of 10 and has a trailing zero.

Special Thanks: Samia Haque Tisha.

Source Code:

```c
#include <stdio.h>
int main(){
    int n;
    scanf("%d" , &n);
    if(n%10 == 0)printf("Yes\n");
    else printf("No\n");

    return 0;
}
```

Time complexity: O(1)

Problem C: Meal Rate

Problem Setter: Sourav Halder
Reviewer: Md. Albin Hossain
Alternate Writer: Samia Haque Tisha

Problem Analysis: The problem is basically to do simple multiplication and division.

Step-1: Multiplication
N bachelors live in a flat and each of them eats MM meals in a month. Therefore, the total meal will be N×M which can be found by doing multiplication of M and N.

Step-2: Division
The bachelors buy groceries of T taka in a month for cooking the meals. Therefore, meal rate or the average cost per meal will be T/(N*M) which can be found by doing a division.

Special Thanks: My heartiest thanks to 'Albin Bhai' for reviewing my problem and helping me a lot.

Source Code:

```c
#include <stdio.h>
int main() {
    int n, m, t;
    scanf("%d %d %d", &n, &m, &t);
    int meal_rate = t / (n * m);
    printf("%d\n", meal_rate);
    return 0;
}
```

Time complexity: O(1)

Problem D: Score Points by Finding Score

Problem Setter: Samia Haque Tisha
Reviewer: Sabbir Ahamed Mridha
Alternate Writer: Sabbir Ahamed Mridha

Observation: Just adding winning points of each arena sequentially does the job of finding total_points. If Turo passes all the arenas only then he wins the match.
Solution Idea: Initially the total_points of Turo is 0. Now, you only need to write nested if statements to check if he passes all arenas or not to declare win or lose.

- If Turo takes less than or equal 35 seconds time in the first arena, 75 points add to total_points. Otherwise he will lose the match and have 0 total_points.
- If he passes the first arena and moves to the next arena if Turo takes less than or equal 45 seconds time in the second arena. If he does then 150 points add to total_points and wins the match. Otherwise he will lose the match because of losing the second arena and his total_points will be 75 points.

Special Thanks: Sabbir Ahamed Mridha

Source Code:

```c
#include <stdio.h>

int main(){
    int t1, t2, total_points=0;
    scanf("%d %d", &t1, &t2);
    if(t1 <= 35){
        total_points += 75;
        if(t2 <= 45){
            total_points += 150;
            printf("Win\n");
            printf("%d\n", total_points);
        }
        else{
            printf("Lose\n");
            printf("%d\n", total_points);
        }
    }
    else{
        printf("Lose\n");
        printf("%d\n", total_points);
    }
    return 0;
}
```

Time complexity: O(1)

Problem E: Cursed Numbers

Problem Setter: MD. Fahim Istiak Nabil
Reviewer: Sabbir Ahamed Mridha
Alternate Writer: Sabbir Ahamed Mridha

Let's traverse through 11 to nn. If i ($1 \le i \le n$) is divisible by xx, and the respected flag of ii is 11, then ii is the particular number that we are looking for.

Now all we have to do is, counting all the numbers that meet both conditions. That's the straightforward answer.

Source Code:

```c
#include<stdio.h>
int main()
{
    int n, x, result = 0;
    scanf("%d %d", &n, &x);
    for (int i = 0; i < n; i++) {
        int in;
        scanf("%d", &in);
        if (in && ((i + 1) % x == 0))result++;
    }
    printf("%d\n", result);
}
```

Time complexity: O(n)

Problem F: Division Tivision

Problem Setter: Sabbir Ahamed Mridha
Reviewer: MD. Fahim Istiak Nabil
Alternate Writer: MD. Fahim Istiak Nabil

Problem Analysis:

- Observation: Idea for this problem is pretty straight forward. Looking at constraints you can solve it in many ways. So all you have to find out is the maximum power of two which divides a number.
- Solution Idea 1: Initial idea for this problem is to multiply a and b and divide it by 2 until it can be divided (x). Same goes for c and d. First we multiply and then divide it by 2 until it can be divided (y). Then we will check whether x is divisible by y or not. That does our job. There are some basic small points that have to be noted to solve this problem with this idea. If you use int data type any multiply, it will lead you to overflow because the constraints for this problem is 10^6 each number. So you must use a long long data type. If you want to avoid long long, then you can divide a and b individually and then add up the powers for nominator and same for denominator. Another case which will lead you to TLE is, you have to take extra care of a and b when at least one of them is zero.
- Solution Idea 2: There is a property, the highest power of two that divides a number is the number of zeros in its binary representation. Number of twos in 7 is zero(111), the number of twos in 6 is one(110) and the number of zeros in 8 is three(1000). Special 0 has zero twos in its binary representation.

So you can find out the number of zeros in a numbers binary representation and just check whether the nominator is divisible by denominator. You can find the number of trailing zeros by a builtin function like (__builtin_ctz(x)).
Special Thanks: Fahim Istiak Nabil / Albin Hossain

Source Code:

```
#include<stdio.h>
int main(){
    int a, b, c, d, twosUp = 0, twosDown = 0;
    scanf("%d %d %d %d", &a, &b, &c, &d);
    long long int nominator = ((long long)(a)) * b;
    long long int denominator = ((long long)(c)) * d;
    if (nominator) {
        while (nominator % 2 == 0) {
            twosUp++;
            nominator /= 2;
        }
    }
    while (denominator % 2 == 0) {
        twosDown++;
        denominator /= 2;
    }
    if (twosUp % twosDown == 0) printf("YES\n");
    else printf("NO\n");
}
```

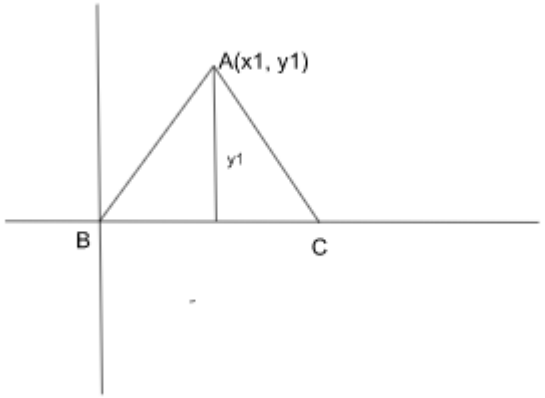
Time complexity: $O(\log_2(\max(a*b, c*d)))$

Problem G: Secret Treasure

Problem Setter: MD. Fahim Istiak Nabil
Reviewer: Md. Albin Hossain
Alternate Writer: Md. Albin Hossain, Sabbir Ahamed Mridha

Problem Analysis:

1. Point B is the origin. So B(0,0).
2. As B and C are both located on the X-axis, hence y1 is the obvious height of the triangle.



Solution Idea :
From these two pieces of information, we can find out the length of BC using the triangle area formula.
ABC = ½ (BC x y1)
BC = (ABC/ y1) * 2
Hence coordinate of C is (BC, 0)
Using the Euclidean distance formula, we can find the length AC and AB.
Now we know the length AB, BC, and AC.
Hence, the circumradius r = (AB*BC*AC)/(4*ABC)
Diameter = 2*r.

Source Code:

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

const double EPS = 1e-7;

struct point {
    double x, y;
    point() { x = y = 0.0; }
    point(double _x, double _y) : x(_x), y(_y) {}
};

double dist(point p1, point p2){ return hypot(p1.x - p2.x, p1.y - p2.y); }

double Area(double b, double h) { return 0.5 * b * h; }

double area;
double rCircumCircle(double ab, double bc, double ca) { return ab * bc * ca / (4.0 *
area); }

double dCircumCircle(point a, point b, point c) { return 2.0 * rCircumCircle(dist(a, b),
dist(b, c), dist(c, a)); }

int main() {
    point A, B, C;
    int tests, cases = 1; cin >> tests;
    assert(tests >= 1 && tests <= 100);
```

```
    while (tests--) {
        cin >> A.x >> A.y >> area;
        C.x = (2 * area / abs(A.y));
        printf("Case %d: %.10lf\n", cases++, dCircumCircle(A, B, C) + EPS);
    }

    return 0;
}
```

Time complexity: O(1)

Problem H: Anagrams

Problem Setter: Md. Albin Hossain
Reviewer: MD. Fahim Istiak Nabil
Alternate Writer: MD. Fahim Istiak Nabil

For this problem we have to check if there are at least two substrings of **s** having the same character frequencies and we have to find the length of the largest of such substrings. Now observe that, if a character c first occurred at position **x** and last occurred at position **y** then we can make two substrings with similar character frequency with length **(y-x)-1** including characters between **x** and **y** and **x**-th character for first substring and **y**-th character for second substring.

For example, from the string “abcabc”, for ‘b’ we can make ”bca” and ”cab” which have the same character frequency.

We find such lengths for all characters from ‘a’ to ‘z’ and maximum of them is our desired length.

Source Code:

```
#include "stdio.h"
#include "string.h"
#define max(a, b) (a > b? a : b)
#define min(a, b) (a < b? a : b)

char s[1000006];
int i, tests, s_length, max_length, mx[26], mn[26];

int main(int argc, char* argv[]) {
    scanf("%d", &tests);
    while (tests--) {
        scanf("%s", s);
        s_length = strlen(s), max_length = 0;

        for (i = 0; i < 26; i++)mx[i] = -1, mn[i] = s_length + 1;

        for (i = 0; i < s_length; i++) {
            mx[s[i] - 'a'] = max(mx[s[i] - 'a'], i);
            mn[s[i] - 'a'] = min(mn[s[i] - 'a'], i);
        }
        for (i = 0; i < 26; i++) max_length = max(max_length, mx[i] - mn[i]);

        printf("%d\n", max_length);
    }
    return 0;
}
```

Time complexity: O(**|S|**)