Judging Director: Abu Saleh
Associate Judging Director: Masum Bin Hossain
Slot B Judging Director: Riadh Hasan
Slot B Associate Judging Director: Md Rana Hossain

Problem A:

Problem Setter : Riadh Hasan
Reviewer: Nusrat Jahan Nory & Md Safiqul Islam Nasim
Alternate Writer:
Analysis:

We just have to print “You can not solve a problem until you are asking the right question.” **as written in the given code.**

Solution code:

```
#include <stdio.h>
int main()
{
    printf("You can not solve a problem until you are asking the right question.\n");
}
```

Time complexity: O(1)

Problem B:

Problem Setter : Nusrat Jahan Nory
Reviewer: Md Safiqul Islam Nasim
Alternate Writer:
Analysis:

This is a simple math problem. In this problem, we have a current rating which is 1535. Now we have to take an input and add these two numbers. Here we have to add two numbers, that's it.

Solution code:

```
#include<stdio.h>
int main()
{
    int A,B;
    scanf("%d %d",&A,&B);
    if(A>B)
    {
     printf("Andy is Happy.\n");
    }
```

```
    else
    {
     printf("Enan is Happy.\n");
    }
   return 0;
}
```

Time complexity: O(1)
Space complexity: O(1)

Problem C:

Problem Setter : Md Safiqul Islam Nasim
Reviewer: Mahir Hasan Sifat
Alternate Writer: Nusrat Jahan Nory
Analysis:

Observation: Here given 2 integers and have to find which number is greater.
Solution Idea: Print “Andy is Happy.” if the first number is greater than the second number, and print “Enan is Happy.” if the second number is greater than the first number.
Solution code:

```
#include <stdio.h>
int main()
{
    int a = 1535;
    int n ;
    scanf("%d" , &n);
    printf("%d\n", n + a);
    return 0;
}
```

Time complexity: O(1)
Space complexity: O(1)

Problem D:

Problem Setter : Mahir Hasan Sifat
Reviewer: Md Rana Hossain
Alternate Writer:
Analysis:

Problem states that there will be given two Integer A and B, and B will be always less than A, B<A. Now, If A is greater than 12 and B is even, means B%2=0, B can be evenly divisible by 2 then you have to print A-B. but if B is not evenly divisible by 2 means odd

(B%2=1), then you have to print A+B.now let's take a look at the other two conditions,If A is less than or equal to 12 and B is even, then you have to print A+B. but if B is not evenly divisible by 2 means odd, then you have to print A-B.

Solution code:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int a,b;
    cin>>a>>b;

    if (a>12 and b%2==0)
    {
        cout<<a-b<<endl;
    }
    else if (a>12 and b%2)
    {
        cout<<a+b<<endl;
    }
    else if (a<=12 and b%2==0)
    {
        cout<<a+b<<endl;
    }
    else
    {
        cout<<a-b<<endl;
    }

}
```

Time complexity: O(1)
Space complexity: O(1)

Problem E:

Problem Setter : Md Rana Hossain
Reviewer: Muhammad Nazmus Sakib
Alternate Writer: Md Safiqul Islam Nasim
Analysis:

Here we have to count the total number of contestants who got the top 300 rank in the preliminary round of the **Ac_Hobe** programming contest.so for the answer you have to traverse the array from start to end and count the value which is less than or equal to 300.

Solution code:

```
#include <stdio.h>
int main()
{
    int n, i, j;
    scanf("%d", &n);
    int a[n + 1], ans = 0;
    for (i = 1; i <= n; i++)
    {
        scanf("%d", &a[i]);
        if (a[i] <= 300)
            ans++;
    }
    printf("%d\n", ans);
}
```

Time complexity: O(n)
Space complexity: O(n)

Problem F:

Problem Setter : Md Safiqul Islam Nasim
Reviewer: Mahir Hasan Sifat
Alternate Writer: Md Rana Hossain
Analysis:

In this problem we have a string that is a wrong-pronounced word then we have some characters that when we pronounced that word correctly that time those characters are silent in that pronunciation. So we have to print that word (string) without those silent characters.

To solve the problem we will traverse every character. Now when we traverse a character we will search for that character in the non-pronounce character list if we find that character we will skip this character and move to the next one. Otherwise, we will print that character. After traversing the full string we will have our answer.

Solution code:

```
#include <stdio.h>
```

```
#include <string>
int main()
{
    int tt;
    scanf("%d", &tt);
    for (int t = 1; t <= tt; t++)
    {
        int n, m;
        scanf("%d %d", &n, &m);
        char st[10000];
        scanf("%s", &st);
        char ch[10000];
        for (int i = 0; i < m; i++)
        {
            char c;
            getchar();
            scanf("%c", &c);
            ch[i] = c;
        }
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < m; j++)
            {
                if (st[i] == ch[j])
                {
                    st[i] = '1';
                }
            }
        }
        for (int i = 0; i < n; i++)
        {
            if (st[i] == '1')
                continue;
            else
                printf("%c", st[i]);
        }
        printf("\n");
    }
}
```

Time complexity: O(T*n*m)

Space complexity: O(n+m)

Problem G:

Problem Setter : Muhammad Nazmus Sakib
Reviewer: Riadh Hasan
Alternate Writer:
Analysis:

As **AC** is the Diagonal of the Rectangle ABCD the absolute difference between x1 and x2 will be rectangle length(l) and the absolute difference between absolute y1 and y2 will be rectangle width(d). As we also know the biggest circle we can draw inside a rectangle is that's diameter will be equal to the minimum value of the rectangle's length(l) and width(d). So now we have the diameter of the circle.
Now as the triangle **ΔEFG is the right triangle** we know only the intersection of two diameters in a circle makes a right angle we can say EF and FG is equal to half of our circle diameter. So now applying the Pythagoras formula we can find the value of EG.

Solution code:

```
#include <bits/stdc++.h>
using namespace std;
void solve()
{

    double x1, x2, y1, y2;
    cin >> x1 >> y1 >> x2 >> y2;
    double gap1 = abs(x1 - x2);
    double gap2 = abs(y1 - y2);
    double R = min(gap1, gap2);
    double r2 = R / 2;
    double acs = (r2 * r2) + (r2 * r2);
    double ans = sqrt(acs);
    printf("%.10lf\n", ans);
}

int main()
{

    int n;
    cin >> n;
    while (n--)
```

```
        solve();

    return 0;
}
```

Time complexity: O(T)
Space complexity: O(1)

Problem H:

Problem Setter : Md Rana Hossain
Reviewer: Muhammad Nazmus Sakib
Alternate Writer: Md Safiqul Islam Nasim
Analysis:

In this problem, you have an array and you have to count how many triplets can you make from that array such that **a[i]** , **a[j] and a[k]** where **(1 <= i < j < k <= N)** hold and the three values of the triplet become prime numbers.

The main idea for this problem is that we don't need all numbers under the square root value of a number to check whether it is prime or not. We just calculate all prime numbers under the square root of a number and check whether that number is divisible by those prime numbers or not. If this number is divisible by any calculated prime number then we can say that this number is not prime otherwise it is a prime number.

Now, in that problem, our maximum array value is 10^9. Its square root value is 32000. So first we just precalculate all prime numbers under 32000 which is approximately 3400 Then we just traverse the array and check every element of the array whether this element is divisible by any of those prime numbers or not. If it is divisible we consider it not a prime number otherwise we consider it a prime number. In that way, we can calculate how many prime numbers are in the given array.

Now, we can calculate the total number of prime numbers in our array. Now, we can select triplets of prime numbers in **nC3 ways**. Here n is the count of prime numbers in our array.

Here nC3 = **(n*(n-1)*(n-3))/6**. From the equation of nCr.

The final answer is **((((n*(n-1))*(n-3))))/6**.

Solution code:

```
#include<stdio.h>
int mod=1000000007;
bool check_prime(long long int n)
{
```

```
    for(int i=2;i*i<=n;i++)
    {
        if(n%i==0)
            return false;
    }
    return true;
}
int main()
{
    long long int t,cas=1,i;
    scanf("%lld",&t);
    long long int prime[50000],j=0;
    for(i=2;i<=32000;i++)
    {
        if(check_prime(i))
        {
            prime[j]=i;
            j++;
        }
    }
    int size=j;
    while(t--)
    {
        long long int n,i,j;
        scanf("%lld",&n);
        long long int a[n+1],cnt=0;
        for(i=1;i<=n;i++)
        {
            scanf("%lld",&a[i]);
            if(a[i]==1)
                continue;
            int flag=0;
            for(j=0;j<size&&prime[j]*prime[j]<=a[i];j++)
            {
                if(a[i]%prime[j]==0)
                {
                    flag=1;
                    break;
                }
            }
```

```
            if(flag==0)
                cnt++;
        }
        printf("Case %lld:
%lld\n",cas++,((((cnt*(cnt-1))%mod)*(cnt-2))%mod)/6);
    }

}
```

Time complexity: O(T*N*3400)
Space complexity: O(N)