LAB IMPLEMENTATION RECORD

COMPUTER PROGRAMMING USING C

BCSC 0851

Institute of Engineering & Technology

B. TECH(CS/VLSI) HONORS
(2021-22)

Submitted By

Shiva Srivastava

(2115800023)

Submitted To

Ms. Harvinder Kaur

Senior Trainer

Department of Computer Engineering & Applications

Department of Computer Engineering and Applications,
GLA University, 17 km. Stone NH#2,
Mathura-Delhi Road, Chaumuha, Mathura – 281406, UP

CERTIFICATE

This is to certify that I, **Shiva Srivastava**, have done hands-on implementation of all the practical's mentioned in this Lab Implementation Record File under the supervision of **Ms. Harvinder Kaur**, **Senior Trainer, Training and Development Department.** I have submitted the file in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (Computer Science & Engineering) Honors.

Signature of Supervisor:

Name of Supervisor: Ms. Harvinder Kaur, Senior Trainer

Date: 20 December 2021

TABLE OF CONTENTS

PRACTICAL NO. 1
OPERATORS AND EXPRESSIONS

1. 1 Adding two numbers

Problem Statement

Every problem starts with a Problem Statement. It tells you in detail about the task to be solved. Usually, although not necessarily, it is accompanied with a story. As a competitive programmer, it is your job to break the problem statement and figure out exactly what it is asking.
Shivam is the youngest programmer in the world, he is just 12 years old. Shivam is learning programming and today he is writing his first program.

The task is very simple: given two integers A and B, write a program to add these two numbers and output it.

Input

This section tells you the format in which your program should receive the input.
The first line contains an integer T, the total number of test cases. Then follow T lines, each line contains two Integers A and B.

Output

This section tells us the format in which your program should give the
output For each test case, add A and B and display the sum in a new line.

Take special care for the output format; everything your program prints is considered “output”, so if you output some debugging statements like “Please enter T” or print something like: “The answer is: ”, this will be considered as part of your answer, and because it does not satisfy the output format, it will be marked wrong, even if your answer is otherwise correct!

Constraints

This section tell you the maximum and minimum possible values the variables in the problem statement can take. You do not need to check these constraints in your program. You can safely assume that the input given to your program will be in the given range of constraints.
$1 \leq T \leq 1000$
$0 \leq A,B \leq 10000$

Sample Input 1
3
1 2

100 200
10 40

Sample Output 1
3
300
50

Solution: #include <stdio.h>

```
int main() {
    // Read the number of test cases.
    int T;
    scanf("%d", &T);
    while (T--) {
            // Read the input a, b
            int a, b;
            scanf("%d %d", &a, &b);

            // Compute the ans.
            int ans = a + b;
            printf("%d\n", ans);
    }

    return 0;
}
```

Result:

main.c

```c
#include <stdio.h>

int main() {
    // Read the number of test cases.
    int T;
    scanf("%d", &T);
    while (T--) {
        // Read the input a, b
        int a, b;
        scanf("%d %d", &a, &b);

        // Compute the ans.
        int ans = a + b;
        printf("%d\n", ans);
    }

    return 0;
}
```

```
2
5 7
12
3 9
12
```

1.2 Find Remainders

Problem statement

Write a program to find the remainder when an integer A is divided by an integer B.

Input

The first line contains an integer T, the total number of test cases. Then T lines follow, each line contains two Integers A and B.

Output

For each test case, find the remainder when A is divided by B, and display it in a new line.

Constraints

$1 \leq T \leq 1000$

$1 \leq A,B \leq 10000$

Example

Input

3

1 2

100 200

40 15

Output

1

100

10

Solution:

```
#include <stdio.h>

int main() {
        // Read the number of test cases.
        int T;
        scanf("%d", &T);
        while (T--) {
```

```
        // Read the input a, b
        int a, b;
        scanf("%d %d", &a, &b);

        // Compute the ans.
        // Complete this line.
        int ans = a%b;
        printf("%d\n", ans);
    }

    return 0;
}
```

Result:

```c
#include <stdio.h>

int main() {
    // Read the number of test cases.
    int T;
    scanf("%d", &T);
    while (T--) {
        // Read the input a, b
        int a, b;
        scanf("%d %d", &a, &b);

        // Compute the ans.
        // Complete this line.
        int ans = a%b;
        printf("%d\n", ans);
    }

    return 0;
}
```

```
3
7 3
1
4 2
0
8 4
0
```

1.3 **First and Last Digit**

Problem Statement

If Give an integer N . Write a program to obtain the sum of the first and last digits of this number.

Input

The first line contains an integer T, the total number of test cases. Then follow T lines, each line contains an integer N.

Output

For each test case, display the sum of first and last digits of N in a new line.

Constraints

$1 \leq T \leq 1000$

$1 \leq N \leq 1000000$

Example

Input

3

1234

124894

242323

Output

5

5

Solution:

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

```
int main()
{int T,s,first,last,i;  // T is number of test cases , s is given number
scanf("%d",&T);
for(i=1;i<=T;i++) // first is first digit and last is last digit
{scanf("%d",&s);
last=s%10;
while(s!=0)
{first=s;
s=s/10;}
if(s==0)
printf("%d\n",last+first);}
return 0;}
```

Result:

```c
#include<stdio.h>
int main()
{int T,s,first,last,i;  // T is number of test cases , s is given number
scanf("%d",&T);
for(i=1;i<=T;i++) // first is first digit and last is last digit
{scanf("%d",&s);
last=s%10;
while(s!=0)
{first=s;
s=s/10;}
if(s==0)
printf("%d\n",last+first);}
return 0;}
```

input

```
2
4567
11


2345
7
```

1.4 Mahasena

Problem Statement

Kattapa, as you all know was one of the greatest warriors of his time. The kingdom of Maahishmati had never lost a battle under him (as army-chief), and the reason for that was their really powerful army, also called as Mahasena.

Kattapa was known to be a very superstitious person. He believed that a soldier is "lucky" if the soldier is holding an even number of weapons, and "unlucky" otherwise. He considered the army as "READY FOR BATTLE" if the count of "lucky" soldiers is strictly greater than the count of "unlucky" soldiers, and "NOT READY" otherwise.

Given the number of weapons each soldier is holding, your task is to determine whether the army formed by all these soldiers is "READY FOR BATTLE" or "NOT READY".

Note: You can find the definition of an even number here.

Input

The first line of input consists of a single integer N denoting the number of soldiers. The second line of input consists of N space separated integers A1, A2, ..., AN, where Ai denotes the number of weapons that the ith soldier is holding.

Output

Generate one line output saying "READY FOR BATTLE", if the army satisfies the conditions that Kattapa requires or "NOT READY" otherwise (quotes for clarity).

Constraints

$1 \leq N \leq 100$

$1 \leq Ai \leq 100$

Example 1

Input:

1

1

Output:

NOT READY

Example 2

Input:

1

2

Output:

READY FOR BATTLE

Example 3

Input:

4

11 12 13 14

Output:

NOT READY

Example 4

Input:

3

2 3 4

Output:

READY FOR BATTLE

Example 5

Input:

5

1 2 3 4 5

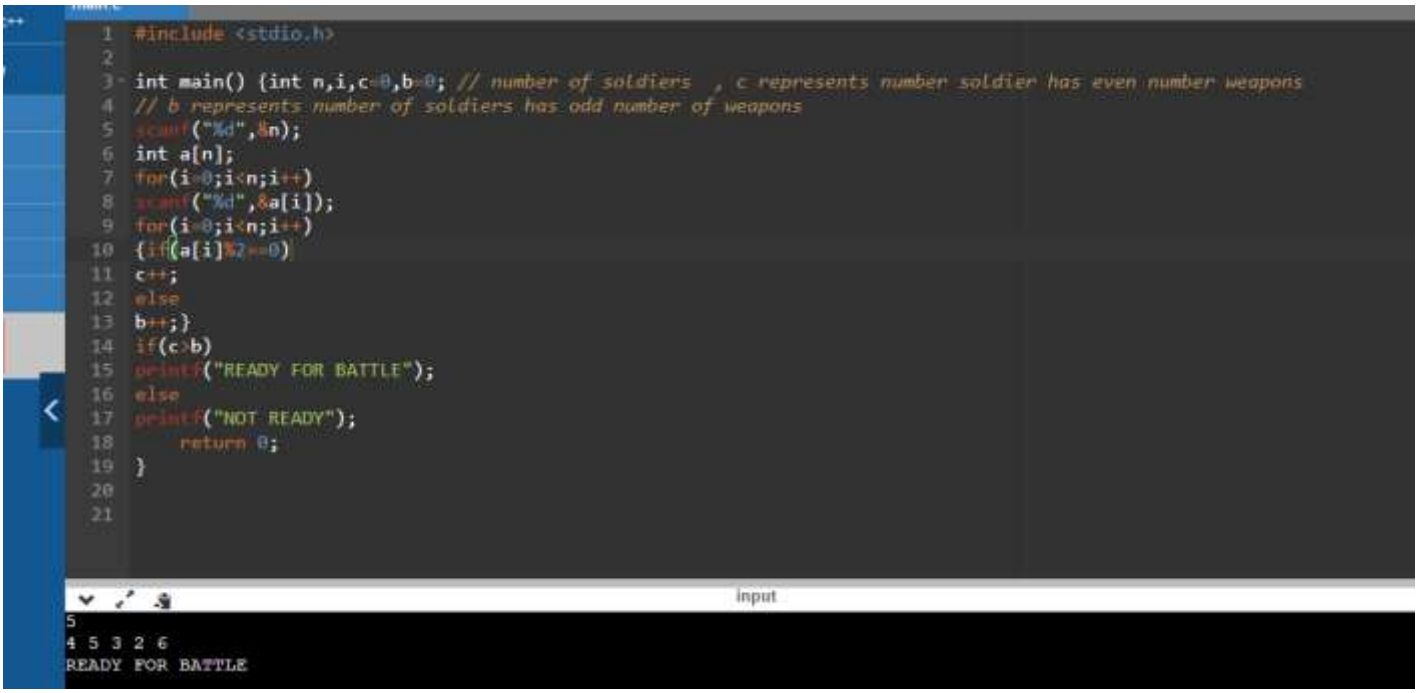
Output:

NOT READY

Solution:

```
#include <stdio.h>

int main() {int n,i,c=0,b=0; // number of soldiers  , c represents number soldier has
even number weapons
// b represents number of soldiers has odd number of weapons
scanf("%d",&n);
int a[n];
for(i=0;i<n;i++)
scanf("%d",&a[i]);
for(i=0;i<n;i++)
{if(a[i]%2==0)
c++;
else
b++;}
if(c>b)
printf("READY FOR BATTLE");
else
printf("NOT READY");
        return 0;
}
```



```c
#include <stdio.h>

int main() {int n,i,c=0,b=0; // number of soldiers  , c represents number soldier has even number weapons
// b represents number of soldiers has odd number of weapons
scanf("%d",&n);
int a[n];
for(i=0;i<n;i++)
scanf("%d",&a[i]);
for(i=0;i<n;i++)
{if(a[i]%2==0)
c++;
else
b++;}
if(c>b)
printf("READY FOR BATTLE");
else
printf("NOT READY");
    return 0;
}
```

Input

```
5
4 5 3 2 6
READY FOR BATTLE
```

1.5 **Second Max of Three Numbers**

Problem Statement

Write a program that accepts sets of three numbers, and prints the second-maximum number among the three.

Input

First line contains the number of triples, N.

The next N lines which follow each have three space separated integers.

Output

For each of the N triples, output one new line which contains the second-maximum integer among the three.

Constraints

$1 \leq N \leq 6$

$1 \leq$ every integer ≤ 10000

The three integers in a single triplet are all distinct. That is, no two of them are equal.

Sample Input

3

1 2 3

10 15 5

100 999 500

Sample Output

2

10

500

Solution:

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

```
int main() {
int a,b,c,N,i;
// n is total test cases
scanf("%d",&N);
for(i=1;i<=N;i++)
{scanf("%d%d%d",&a,&b,&c);
// a is one element , b is second element  & c is third element
if((a>b)&&(b>c))
printf("%d\n",b);
else if((a>c)&&(c>b))
printf("%d\n",c);
else if ((b>c)&&(c>a))
printf("%d\n",c);
else if((b>a)&&(a>c))
printf("%d\n",a);
else if ((c>b)&&(b>a))
printf("%d\n",b);
else if ((c>a)&&(a>b))
printf("%d\n",a);}
return 0;
}
```

Result:

main.c

```c
#include <stdio.h>

int main() {
int a,b,c,N,i;
// n is total test cases
scanf("%d",&N);
for(i=1;i<=N;i++)
{scanf("%d%d%d",&a,&b,&c);
// a is one element , b is second element  & c is third element
if((a>b)&&(b>c))
printf("%d\n",b);
else if((a>c)&&(c>b))
printf("%d\n",c);
else if ((b>c)&&(c>a))
printf("%d\n",c);
else if((b>a)&&(a>c))
printf("%d\n",a);
else if ((c>b)&&(b>a))
printf("%d\n",b);
else if ((c>a)&&(a>b))
printf("%d\n",a);}
    return 0;
}
```

input

```
2
5 4 2
4
594 3 1234
594
```

1.6 Greedy puppy

Problem statement

Tuzik is a little dog. But despite the fact he is still a puppy he already knows about the pretty things that coins are. He knows that for every coin he can get very tasty bone from his master. He believes that some day he will find a treasure and have loads of bones.

And finally he found something interesting. A wooden chest containing N coins! But as you should remember, Tuzik is just a little dog, and so he can't open it by himself. Actually, the only thing he can really do is barking. He can use his barking to attract nearby people and seek their help. He can set the loudness of his barking very precisely, and therefore you can assume that he can choose to call any number of people, from a minimum of 1, to a maximum of K.

When people come and open the chest they divide all the coins between them in such a way that everyone will get the same amount of coins and this amount is maximal possible. If some coins are not used they will leave it on the ground and Tuzik will take them after they go away. Since Tuzik is clearly not a fool, he understands that his profit depends on the number of people he will call. While Tuzik works on his barking, you have to find the maximum possible number of coins he can get.

Input

The first line of the input contains an integer T denoting the number of test cases. Each of next T lines contains 2 space-separated integers: N and K, for this test case.

Output

For each test case output one integer - the maximum possible number of coins Tuzik can get.

Constraints

- $1 \leq T \leq 50$
- $1 \leq N, K \leq 105$

Sample Input 1

2

5 2

11 3

Sample Output 1

1

2

Solution:

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

```
int main() {
int n,t,m,i,j,max;
// max is maximum value which come  from 1 to m cases
scanf("%d",&m);
// m is total test cases
for(i=1;i<=m;i++)
{
scanf("%d%d",&n,&t);
//n is number of coins
// t is max bark by dog
max=0;
for(j=1;j<=t;j++)
  {
   if(max<n%j)
   max=n%j;}
   printf("%d\n",max);
}

    return 0;
}
```

Result:

```c
#include <stdio.h>

int main() {
int n,t,m,i,j,max;
// max is maximum value which come  from 1 to m cases
scanf("%d",&m);
// m is total test cases
for(i=1;i<=m;i++)
{
scanf("%d%d",&n,&t);
//n is number of coins
// t is max bark by dog
max=0;
for(j=1;j<=t;j++)
   {
    if(max<n%j)
    max=n%j;}
    printf("%d\n",max);
}

    return 0;
}

```

input

```
2
 10 4
2
23 6
5
```

1.7 GCD AND LCM

Problem Statement

Two integers **A** and **B** are the inputs. Write a program to find GCD and LCM of A and B.

Input

The first line contains an integer **T**, total number of testcases. Then follow **T** lines, each line contains an integer **A** and **B**.

Output

Display the GCD and LCM of **A** and **B** separated by space respectively. The answer for each test case must be displayed in a new line.

Constraints

- $1 \leq \mathbf{T} \leq 1000$
- $1 \leq \mathbf{A,B} \leq 1000000$

Example

Input

3

120 140

10213 312

10 30

Output

20 840

1 3186456

10 30

Solution:

```
#include <stdio.h>

int main() {
int i,a,b,GCD,max,t,j;
scanf("%d",&t);
//GCD is greatest common divisor
```

```
//max is LCM
//a and b  are given two numbers
for(i=1;i<=t;i++)
{scanf("%d%d",&a,&b);
for(j=1;j<=(int)a/2;j++)
{if((a%j==0)&&(b%j==0))
GCD=j;}
printf("%d\t",GCD);
max=(a>b)?a:b;
while(1)
{{if((max%a==0)&&(max%b==0))
{printf("%d\n",max);
break;}}
++max;} }


return 0;
}
```

```c

int main() {
int i,a,b,GCD,max,t,j;
scanf("%d",&t);
//GCD is greatest common divisor
//max is LCM
//a and b  are given two numbers
for(i=1;i<=t;i++)
{scanf("%d%d",&a,&b);
for(j=1;j<=(int)a/2;j++)
{if((a%j==0)&&(b%j==0))
GCD=j;}
printf("%d\t",GCD);
max=(a>b)?a:b;
while(1)
{{if((max%a==0)&&(max%b==0))
{printf("%d\n",max);
break;}}
++max;} }


    return 0;
```

```
1
40 10
10      40
```

1.8 **Simple Division**

Problem Statement

Given an array A of N integers and two integers X and Y, find the number of integers in the array that are both less than or equal to X and divisible by Y.

Input

The first line of the input contains an integer T denoting the number of test cases. The description of each test case follows.
The first line of each test case contains three space separated integers: **N, X** and **Y**.
The second line contains N space-separated integers **A1, A2, ..., AN** denoting the array A.

Output

For each test case, output a single line containing the answer.

Constraints

- $1 \leq T \leq 10$
- $1 \leq N \leq 10^5$
- $1 \leq A_1, A_2, \ldots, A_N \leq 10^9$
- $1 \leq X, Y \leq 10^9$

Information to score partial points

For 20% of the score, it is guaranteed that $N \leq 100$.

For further 30% of the score, it is guaranteed that $N \leq 1000$.

For the rest of the 50% of the score, no extra guarantees. That is, $N \leq 105$.

Example

Input 1:

1

3 2 1

1 2 3

Output 1:

2

Input 2:

2

5 10 3

4 6 12 3 9

1 10 10

20

Output 2:

3

0

Solution:

```
#include<stdio.h>
int main()
{int t,i,j,k;
scanf("%d",&t);
// t is  total test cases
for(i=0;i<t;i++)
{int n,x,y,m=0;
// n total number of array
// x is number for which less than number
// y is number which divide given number completely
// m is total such cases


   scanf("%d %d %d",&n,&x,&y);
   int a[n];
   for(k=0;k<n;k++)
```

```
    scanf("%d",&a[k]);
    for(j=0;j<n;j++)
    {if((a[j]<x)&&(a[j]%y==0))
    m++;}
    printf("%d\n",m);}
return 0;}
```

1.9 **Alternating Subsequences**

Problem statement

You are given an array of **N** non-negative integers: $\mathbf{A_1}$, $\mathbf{A_2}$, ..., $\mathbf{A_N}$. An *alternating subsequence* is a subsequence in which the indices of any two consecutive elements differ by exactly two in the original array. That is, if $\mathbf{A_{i1}}$, $\mathbf{A_{i2}}$, ..., $\mathbf{A_{ik}}$ is some subsequence, then for it to be an *alternating subsequence*, ($i_2 - i_1 = 2$), ($i_3 - i_2 = 2$), and so on should all hold true. Among all *alternating subsequences*, find the one which has maximum sum of elements, and output that sum.

Input

The first line of the input contains an integer **T** denoting the number of test cases.

The first line of each test case contains an integer **N** denoting the number of elements in the array.

The second line contains **N** space-separated integers $\mathbf{A_1}$, $\mathbf{A_2}$, ..., $\mathbf{A_N}$ denoting the array **A**.

Output

For each test case, output a single line containing the answer.

Note

A subsequence with only a single integer is also an alternating subsequence.

Constraints

- $1 \leq \mathbf{T} \leq 10$
- $1 \leq \mathbf{N} \leq 10^5$
- $0 \leq \mathbf{A_i} \leq 10^5$

Scoring Infomation

- 20% score for $\mathbf{N} \leq 100$
- 30% score for $\mathbf{N} \leq 1000$
- 50% score for $\mathbf{N} \leq 10^5$

Example

Input:

1

3

1 2 5

Output:

6

Solution: #include<stdio.h>

```
 #include<math.h>
 #include<stdlib.h>
int main()
{int t,i,j,n ,sum1,sum2;
scanf("%d",&t);  // t is number of test cases
for(i=1;i<=t;i++)
{scanf("%d",&n);   //n is  length of array
int a[n];
for(j=0;j<n;j++)
{scanf("%d",&a[j]);}
sum1=0;
// sum1 is maximum in even subsequence
for(j=0;j<n-1;j=j+2)
if(sum1<(a[j]+a[j+2])){
sum1=a[j]+a[j+2];}


sum2=0;
// sum2 is maximum in odd subsequence
for(j=0;j<n-1;j=j+2)
if(sum2<(a[j]+a[j+2]))
{sum2=a[j]+a[j+2];}
if(sum1>sum2)
printf("%d\n",sum1);
else
printf("%d\n",sum2);
}
return 0;}
```

Result:

main.c Ctrl+S

```c
  #include<stdio.h>
  #include<math.h>
  #include<stdlib.h>
int main()
{int t,i,j,n ,sum1,sum2;
scanf("%d",&t);  // t is number of test cases
for(i=1;i<=t;i++)
{scanf("%d",&n);   //n is  length of array
int a[n];
for(j=0;j<n;j++)
{scanf("%d",&a[j]);}
sum1=0;
// sum1 is maximum in even subsequence
for(j=0;j<n-1;j=j+2)
if(sum1<(a[j]+a[j+2])){
sum1=a[j]+a[j+2];}


sum2=0;
// sum2 is maximum in odd subsequence
for(j=0;j<n-1;j=j+2)
if(sum2<(a[j]+a[j+2]))
{sum2=a[j]+a[j+2];}
if(sum1>sum2)
printf("%d\n",sum1);
else
printf("%d\n",sum2);
}
return 0;}
```

```
1
3
1 2 5
6


Program finished with exit code 0
```

PRACTICAL NO. 2.
Decision and Case Control structure4

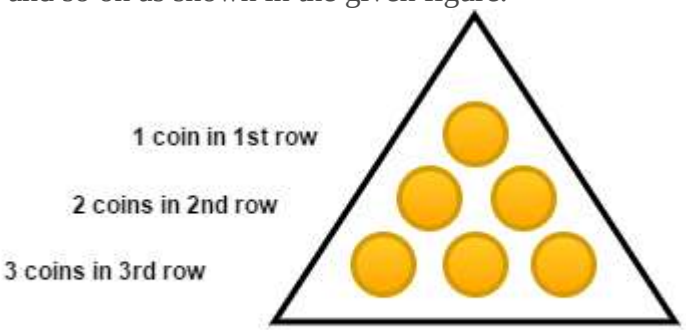
2.1 Coins And Triangle

Problem statement

Chef belongs to a very rich family which owns many gold mines. Today, he brought **N** gold coins and decided to form a triangle using these coins. Isn't it strange?

Chef has a unusual way of forming a triangle using gold coins, which is described as follows:

- He puts **1** coin in the **1st** row.
- then puts **2** coins in the **2nd** row.
- then puts **3** coins in the **3rd** row.
- and so on as shown in the given figure.



A Traingle with height = 3 requires 6 coins

Chef is interested in forming a triangle with maximum possible height using at most **N** coins. Can you tell him the maximum possible height of the triangle?

Input

The first line of input contains a single integer **T** denoting the number of test cases.

The first and the only line of each test case contains an integer **N** denoting the number of gold coins Chef has.

Output

For each test case, output a single line containing an integer corresponding to the maximum possible height of the triangle that Chef can get.

Constraints

- $1 \leq T \leq 100$
- $1 \leq N \leq 10^9$

Subtasks

- Subtask 1 (48 points) : $1 \leq N \leq 10^5$

- Subtask 2 (52 points) : $1 \leq N \leq 10^9$
  Example

Input

3

3

5

7

Output

2

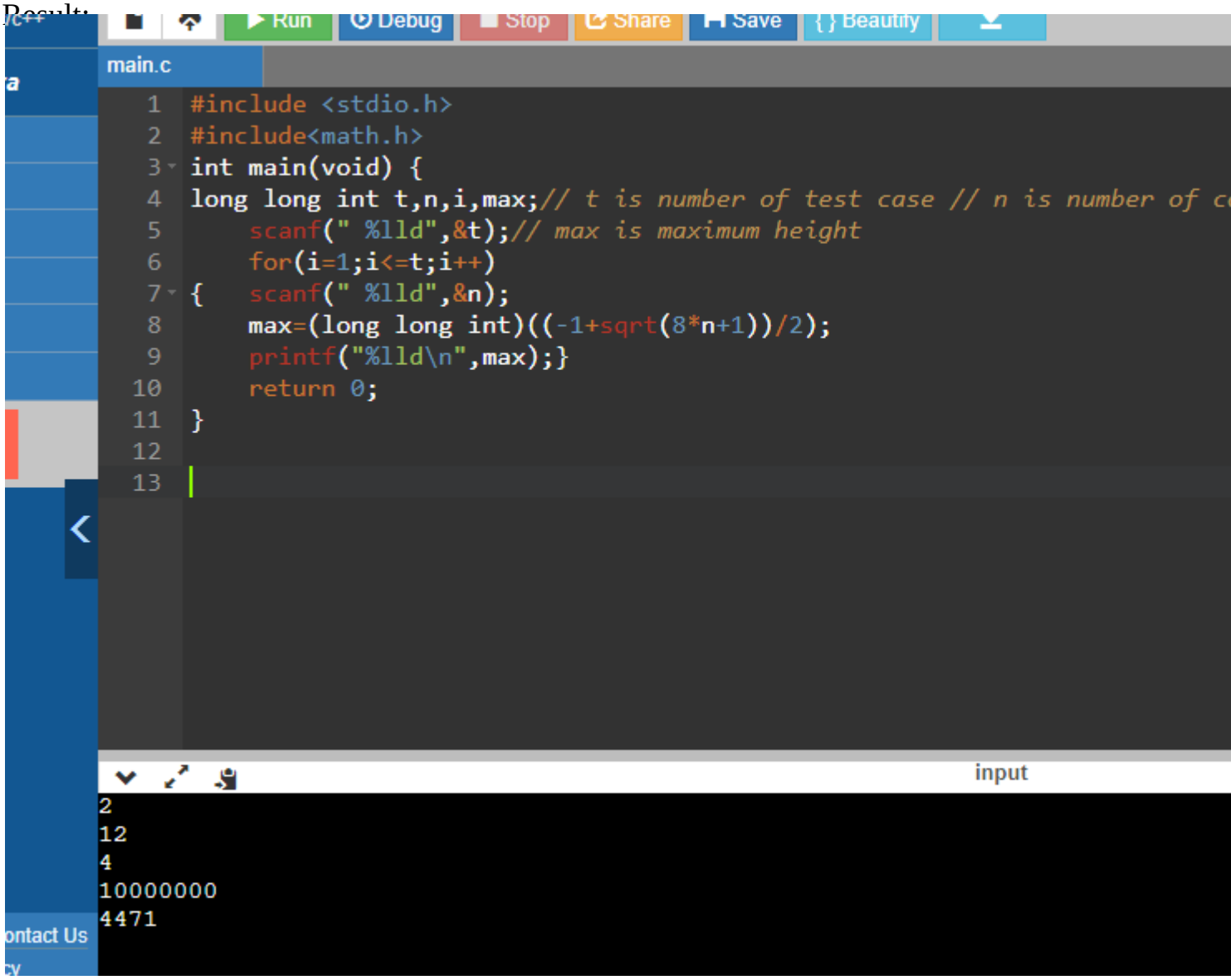
2

3

Solution:

```
#include <stdio.h>
#include<math.h>
long long int main(void) {
    int t,n,i,max;// max is maximum height of triangle formed
    scanf("%lld",&t);// t is total number of test cases
    for(i=1;i<=t;i++)
{   scanf("%lld",&n);// n is total number of coins
   max=(int)((-1+sqrt(8*n+1))/2);
   printf("%lld\n",max);}
    return 0;
}
```

Result:


Run
Debug
Stop
Share
Save
{} Beautify
main.c
#include <stdio.h>
#include<math.h>
int main(void) {
long long int t,n,i,max;// t is number of test case // n is number of c
scanf(" %lld",&t);// max is maximum height
for(i=1;i<=t;i++)
{ scanf(" %lld",&n);
max=(long long int)((-1+sqrt(8*n+1))/2);
printf("%lld\n",max);}
return 0;
}
input
2
12
4
10000000
4471
ontact Us

2.2 Possible Victory

Problem Statement

Chef is playing in a T20 cricket match. In a match, Team **A** plays for 20 overs. In a single over, the team gets to play 6 times, and in each of these 6 tries, they can score a maximum of 6 runs. After Team A's 20 overs are finished, Team **B** similarly plays for 20 overs and tries to get a higher total score than the first team. The team with the higher total score at the end wins the match.

Chef is in Team **B**. Team A has already played their 20 overs, and have gotten a score of **R**. Chef's Team **B** has started playing, and have already scored **C** runs in the first **O** overs. In the remaining **20−O** overs, find whether it is possible for Chef's Team **B** to get a score high enough to win the game. That is, can their final score be strictly larger than R?

Input:

- There is a single line of input, with three integers, **R,O,C**.

Output:

Output in a single line, the answer, which should be **"YES"** if it's possible for Chef's Team B to win the match and **"NO"** if not.

You may print each character of the string in uppercase or lowercase (for example, the strings **"yEs", "yes", "Yes"** and **"YES"** will all be treated as identical).

Constraints

- $0 \leq C \leq R \leq 720$
- $1 \leq O \leq 19$
- $0 \leq C \leq 36 * O$

Sample Input 1:

719 18 648

Sample Output 1:

YES

Explanation:

In the remaining $20-O=2$ overs, Team B gets to play $2*6=12$ times, and in each try, they can get a maximum of 6 score. Which means that the maximum score that they can acheieve in these 2 overs is $12*6=72$. Thus, the maximum total score that Team B can achieve is $C+72=720$. 720 is strictly more than Team A's score of 719, and hence Chef's Team B can win this match.

Sample Input 2:

720 18 648

Sample Output 2:

NO

Solution:

```
#include <stdio.h>

int main(void) {
   int r,o,s;
   scanf("%d %d %d",&r,&o,&s);
   // r is run score by team a
//   printf("%d",s+(20-o)*36);
// o is over is played by team b
//s is score scored in o overs
   if((20-o)*36>(r-s))
   printf("%s","YES");
   else
   printf("%s","NO");

     return 0;
}
```

Result:

```c
#include <stdio.h>

int main(void) {
    int r,o,s;
    scanf("%d %d %d",&r,&o,&s);
    // r is run score by team a
//  printf("%d",s+(20-o)*36);
// o is over is played by team b
//s is score scored in o overs
    if((20-o)*36>(r-s))
    printf("%s","YES");
    else
    printf("%s","NO");

    return 0;
}
```

```
234
4
123
YES
```

2.3 Mutated Minions

Problem Statement

Gru has not been in the limelight for a long time and is, therefore, planning something particularly nefarious. Frustrated by his minions' incapability which has kept him away from the limelight, he has built a transmogrifier — a machine which mutates minions.

Each minion has an intrinsic characteristic value (similar to our DNA), which is an integer. The transmogrifier adds an integer **K** to each of the minions' characteristic value.

Gru knows that if the new characteristic value of a minion is divisible by 7, then it will have Wolverine-like mutations.

Given the initial characteristic integers of **N** minions, all of which are then transmogrified, find out how many of them become Wolverine-like.

Input Format:

The first line contains one integer, **T**, which is the number of test cases. Each test case is then described in two lines.

The first line contains two integers **N** and **K**, as described in the statement.

The next line contains **N** integers, which denote the initial characteristic values for the minions.

Output Format:

For each testcase, output one integer in a new line, which is the number of Wolverine-like minions after the transmogrification.

Constraints:

- **$1 \leq T \leq 100$**
- **$1 \leq N \leq 100$**
- **$1 \leq K \leq 100$**
- All initial characteristic values lie between 1 and 10^5, both inclusive.

Example

Input:

1

5 10

2 4 1 35 1

Output:

1

Solution:

```
#include<stdio.h>
int main()
{int t,n,b=0,i,k,j;//t total cases k for adding
scanf("%d",&t);
for(j=0;j<t;j++){b=0;// b is flag
scanf("%d %d",&n,&k);//n minions
int a[n];
for(i=0;i<n;i++)
{scanf("%d",&a[i]);
if((a[i]+k)%7==0)
b++;}
if(b>=0)
printf("%d\n",b);}
return 0;}
```

Result:

```c
 #include<stdio.h>
int main()
{int t,n,b=0,i,k,j;//t total cases k for adding
scanf("%d",&t);
for(j=0;j<t;j++){b=0;// b is flag
scanf("%d %d",&n,&k);//n minions
// k is added to each element of array
int a[n];
for(i=0;i<n;i++)
{scanf("%d",&a[i]);
if((a[i]+k)%7==0)
b++;}
if(b>=0)
printf("%d\n",b);}
return 0;}
```

```
5
3
2 3 45 4 6
1
```

2.4 Chef and Coupon

Problem Statements

Chef wants to order food from a food delivery app. He wishes to order once today, and buy three items costing A1,A2 andA3 rupees, respectively. He'll also order once tomorrow, when he'll buy three items costing B1,B2,andB3 rupees, respectively. There is an additional delivery charge of rupees D for each order. He notices that there is a coupon on sale, which costs rupees C. If he buys that coupon, the delivery charges on any day, on an order of rupees 150 or more shall be waived (that is, the D rupees will not be added, if the sum of the costs of the items is ≥150).
Note that Chef is ordering the three items together on each day, so the delivery charge is charged only once each day. Also, note that it's only needed to buy the coupon once to avail the delivery fee waiver on both days.

Should Chef buy the coupon? Note that Chef shall buy the coupon only if it costs him **strictly less** than what it costs him without the coupon, in total.

Input:

- The first line of the input contains a single integer T, denoting the number of test cases.
- The first line of each test case contains two space-separated integers D and CC, denoting the delivery charge, and the price of the coupon, respectively.
- The second line of each test case contains three space-separated integers A1,A2A1,A2 and A3A3, denoting the prices of the food items that Chef plans to order on Day 11, respectively.
- The third line of each test case contains three space-separated integers B1B1, B2B2 and B3B3, denoting the prices of the food items that Chef plans to order on Day 22, respectively.

Output:

For each test case, output **YES** if Chef should buy the coupon, and **NO** otherwise, in a separate line.

Constraints

- **1≤T≤104**
- **1≤D,C≤100**
- **1≤A1,A2,A3≤100**
- **1≤B1,B2,B3≤100**

Sample Input 1:

```
2

90 100

100 50 10

80 80 80

30 30

100 100 100

10 20 30
```

Sample Output 1:

YES

NO

Solution:

```
#include<stdio.h>
int main()
{int t,i,d,x,y,z,u,v,w,c;
scanf("%d",&t);
// t is total test cases
for(i=1;i<=t;i++)
{scanf("%d %d ",&d,&c);
//d is delivery charge
//c is coupan prize
scanf("%d %d %d ",&x,&y,&z);
// x,y,z are prize of item buy on first day
scanf("%d %d %d ",&u,&v,&w);
// u,v,w are prize of item buy on first day
if((x+y+z>=150)&&(u+v+w>=150))
{if((x+y+z+c+u+v+w)<(x+y+z+d+d+u+v+w))
printf("YES\n");
else
printf("NO\n");}
else if (x+y+z<150&&u+v+w<150)
printf("NO\n");
else
{
if((x+y+z+c+d+u+v+w)<(x+y+z+d+d+u+v+w))
printf("YES\n");
else
printf("NO\n");
}

}return 0 ;}
```

Custom Input

```
1
30 45
100 100 100
10 20 30
```

Status Successfully executed **Date** 2021-12-19 16:25:09 **Time** 0.003207 sec **Mem** 5.484 kB

Input

```
1
30 45
100 100 100
10 20 30
```

Output

```
NO
```

2.5 Chef on Island

Problem Statements

Suppose Chef is stuck on an island and currently he has xx units of food supply and y units of water supply in total that he could collect from the island. He needs x_r units of food supply and y_r units of water supply per day at the minimal to have sufficient energy to build a boat from the woods and also to live for another day. Assuming it takes exactly D days to build the boat and reach the shore, tell whether Chef has the sufficient amount of supplies to be able to reach the shore by building the boat?

Input:

- First line will contain TT, number of testcases. Then the testcases follow.
- Each testcase contains of a single line of input, five integers x,y,x_r,y,D.

Output:

For each testcase, output in a single line answer "YES" if Chef can reach the shore by building the boat and "NO" if not (without quotes).

You may print each character of each string in uppercase or lowercase (for example, the strings "yEs", "yes", "Yes" and "YES" will all be treated as identical).

Constraints

- $1 \leq T \leq 300$
- $1 \leq x,y \leq 100$
- $1 \leq x_r,y_r,D \leq 10$

Sample Input

3

4 2 1 1 1

4 2 1 3 1

4 2 4 2 2

Sample Output:

YES

NO

NO

Solution:

```
#include<stdio.h>
int main()
{int x,y,u,v,d,t,i;
```

```
scanf("%d",&t);
// t is total test cases
//x is total stock of food
//y is total stock of water
// u is minimum amount of food perday
//v is minimum amout of water reqired perday
//d is  for total days we need to eat that food

for(i=1;i<=t;i++)
{scanf("%d%d%d%d%d",&x,&y,&u,&v,&d);
if((x>=u*d)&&(y>=v*d))
printf("YES\n");
else
printf("NO\n");}
return 0;}
```

Result:

```c
#include<stdio.h>
int main()
{int x,y,u,v,d,t,i;
scanf("%d",&t);
// t is total test cases
//x is total stock of food
//y is total stock of water
// u is minimum amount of food perday
//v is minimum amout of water reqired perday
//d is  for total days we need to eat that food

for(i=1;i<=t;i++)
{scanf("%d%d%d%d%d",&x,&y,&u,&v,&d);
if((x>=u*d)&&(y>=v*d))
printf("YES\n");
else
printf("NO\n");}
return 0;}

```

```
2
4 5
2 3
2
NO
```

2.6 Lazy Chef

Problem Statement

Chef is a very lazy person. Whatever work is supposed to be finished in xx units of time, he finishes it in m∗x units of time. But there is always a limit to laziness, so he delays the work by at max d units of time. Given x,m,d, find the maximum time taken by Chef to complete the work.

Input:

- First line will contain T, number of testcases. Then the testcases follow.
- Each testcase contains a single line of input, three integers x,m,d.

Output:

For each testcase, output in a single line answer to the problem.

Constraints

- $1 \leq T \leq 104$
- $1 \leq x,m \leq 10$
- $0 \leq d < 100$

Sample Input 1

3

1 1 0

1 3 1

2 2 3

Sample Output 1

1

2

4

Solution:

```
#include<stdio.h>
int main()
{int t,x,m,d,i;
// t is total  test cases
scanf("%d",&t);
for (i=1;i<=t;i++)
```

//x is supposed to be when they are not lazy finish on this time

//when they are lazy then may complete it in m*x amout of time

//maximum delays is d units

{scanf("%d%d%d",&x,&m,&d);

if(x*m>x+d)

printf("%d\n",x+d);

else

printf("%d\n",x*m);}

return 0;}

Result:

```c
#include<stdio.h>
int main()
{int t,x,m,d,i;
// t is total  test cases
scanf("%d",&t);
for (i=1;i<=t;i++)
//x is supposed to be when they are not lazy finish on this time
//when they are lazy then may complete it in m*x amout of time
//maximum delays is d units
{scanf("%d%d%d",&x,&m,&d);
if(x*m>x+d)
printf("%d\n",x+d);
else
printf("%d\n",x*m);}
return 0;}
```

input

```
1
4 2 5
8
```