program input output styles

there may be a few kinds of program input styles in contest . we will assume that you are given two numbers and have to show the summation.

style 1:

there will be a test case T given. after that T lines follows. each line contains 2 integer.

example -

```
3
33 12
12 12
10 100
```

say the problems says, output will be like -

```
Case 1: 45
Case 2: 24
Case 3: 110
```

program

```
int main()
{
    int a, b, T, kase;

    for(kase = 1; kase <= T; kase++)
    {
        printf("Case %d: %d\n", kase, a+b );
    }

    return 0;
}
```

***notice that, for the last output (case 3) after 110 also a new line is printed and according to the style there remains a blank line after all the end of output.**

style 2

the number of test cases are not mentioned but an ending indicator is mentioned. say, 2 numbers will be given at each line, but if both of them are 0 0 then the program will terminate (will not produce output for that 0 0 line)

example

```
11 11
22 22
10 10
0 0
```

output will be -

```
22
44
20
```

program

```
int main()
{
    int a, b;

    while( scanf("%d %d", &a, &b) == 2 && a != 0 && b != 0 )
    {
        printf("%d\n", a+b );
    }

    return 0;
}
```

so the program will read while there exists 2 valid inputs but it will not process the line 0 0.

from previous article you know if the end of input is fetched the scanf() returns the value -1 or EOF.

so, you can also write -

```
int main()
{
    int a, b;

    while( scanf("%d %d", &a, &b) != EOF && a != 0 && b != 0 )
    {
        printf("%d\n", a+b );
    }

    return 0;
}
```

this program is run until the end of file or end of input is fetched and not until a and b are both 0.

it can also be written this way -

```
int main()
{
    int a, b;

    while( scanf("%d %d", &a, &b) != EOF )
    {
        if(a == 0 && b == 0)
            break;

        printf("%d\n", a+b );
    }

    return 0;
}
```

so , the program is run for all the inputs. but if both a and b are 0 then it terminates.

***you can also write -1 in place of EOF because value of EOF == -1. do experiment.**

there may be -1 -1 instead of 0 0. then check for those values inside while.

style 3

test case is not given. two values are given in each line. if the line only contains 0 (means the first value is 0) then terminate the program.

example

```
11 11
22 22
10 10
0
```

output

```
22
44
20
```

program

```
int main()
{
    int a, b;

    while( scanf("%d %d", &a, &b) == 2 && a != 0 )
    {
        printf("%d\n", a+b );
    }

    return 0;
}
```

so for the last input line

scanf() is getting only value of a but not b. so scanf() function is not equals 2,

and

a is 0

therefore, both conditions inside while are false. the program will terminate.

you can also write,

```
int main()
{
    int a, b;

    while(scanf("%d %d", &a, &b) != EOF )
    {
        if(a == 0)
            break;

        printf("%d\n", a+b );
    }

    return 0;
}
```

so it checks until the end of input and if a == 0 terminates.

style 4

no test case is given. 2 numbers are given at each line. you have to produce output for all of them.

example -

```
11 11
22 22
10 10
```

output will be

```
22
44
20
```

program

```
int main()
{
    int a, b;

    while(scanf("%d %d", &a, &b) != EOF )
    {
        printf("%d\n", a+b );
    }

    return 0;
}
```

inside while you can also write

```
scanf("%d %d", &a, &b) != -1
```

or,

```
scanf("%d %d", &a, &b) == 2
```

a program with gets

suppose you have a program that says you have to count the given names. each line is a different name and may contain spaces. say length of each name is within 256 characters including spaces.

example

```
ghoseti
pagla dashu
sokhina begum
kan kata ramajan
```

output will be

```
4
```

because there are 4 names.
how to do that program to read any number of names ? you know scanf() can read string but that can not contain space. gets() function can help you for that. the main target of this example is to teach how to use gets() inside while loop.

program

```
int main()
{
    char name[256];
    int c = 0;

    while( gets(str) )
    {
        c++;
    }

    printf("%d", c);

    return 0;
}
```

you can use gets() in this way. because, if a valid string input is found then gets() returns the taken string and that is considered as true. but if EOF is fetched then it returns NULL which is equivalent to 0. so at the end of program input it is automatically terminated.

read the IO.pdf to see more examples about gets().

do not forget to write the line

```
using namespace std
```

immediately after the headers if you are using a C++ project.