Important message on plagiarism

The single most important point for you to realize before the beginning of your studies at ShanghaiTech is the meaning of "plagiarism":

Plagiarism is the practice of taking someone else's work or ideas and passing them off as one's own. It is the misrepresentation of the work of another as your own. It is academic theft; a serious infraction of a University honor code, and the latter is your responsibility to uphold. Instances of plagiarism or any other cheating will be reported to the university leadership, and will have serious consequences. Avoiding any form of plagiarism is in your own interest. If you plagiarize and it is unveiled at a later stage only, it will not only reflect badly on the university, but also on your image/career opportunities.

Plagiarism is academic misconduct, and we take it very serious at ShanghaiTech. In the past we have had lots of problems related to plagiarism especially with newly arriving students, so it is important to get this right upfront:

You may…

• … discuss with your peers about course material.
• … discuss generally about the programming language, some features, or abstract lines of code. As long as it is not directly related to any homework, but formulated in a general, abstract way, such discussion is acceptable.
• … share test cases with each other.
• … help each other with setting up the development environment etc.

You may not …

• … read, possess, copy or submit the solution code of anyone else (including people outside this course or university)!
• … receive direct help from someone else (i.e. a direct communication of some lines of code, no matter if it is visual, verbal, or written)!
• … give direct help to someone else. Helping one of your peers by letting him read your code or communicating even just part of the solution in written or in verbal form will have equal consequences.
• … gain access to another one's account, no matter if with or without permission.
• … give your account access to another student. It is your responsibility to keep your account safe, always log out, and choose a safe password. Do not just share access to your computer with other students without prior lock--out and disabling of automatic login functionality. Do not just leave your computer on without a lock even if it is just for the sake of a 5--minute break.
• … work in teams. You may meet to discuss generally about the material, but any work on the homework is to be done individually and in privacy. Remember, you may not allow anyone to even just read your source code.

With the Internet, "paste", and "share" are easy operations. Don't think that it is easy to hide and that we will not find you, we have just as easy to use, fully automatic and intelligent tools that will identify any potential cases of plagiarism. And do not think that being the original author will make any difference. Sharing an original solution with others is just as unethical as using someone else's work.

CS100 Homework 1 (Spring, 2021)

In this homework, you are required to do some simple programs with C programming language, starting to get familiar with it while solving some simple problems. This homework tries to pave the way for your later homework and projects.

Submission deadline:
2021-03-14 23:59
Late submission will open for 24 hours after the deadline, with -50% point deduction.

Problem 1. Finding Bugs! (30 points)

What is the most frustrating, most time-consuming yet the most crucial part of programming? More likely not writing out your codes but debugging. And that's why this is your first problem!

Your friend has just started learning programming. He/She asks you for help, and sends you this piece of code:

```
int mian()
{
    int numTotalStudents; numSatisfied;
    int satisfiedRatio;

    printf("How many students went to the new dining hall?\n");
    scanf("%d", numTotalStudents);
    printf("How many of them are satisfied?\n");
    scanf("%d", numSatisfied);

    satisfiedRatio = numSatisfied / numTotalStudents

    // Prints different messages according to the ratio!
    if(satisfiedRatio = 0.5f)
    {
        printf("Exactly half of the students are satisfied!\n");
    }
    elseif(satisfiedRatio > 0.5f)
    {
        printf("More students are satisfied! :)\n");
    }
    elseif(satisfiedRatio < 0.5f)
    {
        printf("More students are unsatisfied! :(\n");
    }
    else // In case if it divides by zero
    {
        printf("No one went to the dining hall?\n");
    }
    return 0;
}
```

Please help your friend by discovering all silly mistakes, and making this piece of code work. As your friend did not tell you anything more, you will need to figure out what this program is doing by yourself.

Oh, your friend just sends you a screenshot of OJ! It says a correct program runs like this (red indicates input and black indicates your output):

```
How many students went to the new dining hall?
100
How many of them are satisfied?
80
More students are satisfied! :)
```

```
How many students went to the new dining hall?
0
How many of them are satisfied?
0
No one went to the dining hall?
```

Problem 2. How Is Your Mood Today? (30 points)

As a student in ShanghaiTech, your schedule is filled up with lectures. Some starts as early as 8:15, some ends as late as 21:30. After a day of lectures, sometimes you will feel tired and upset. A same story goes for Gezi Wang, a lazy freshman.

Gezi Wang's day begins when he wakes up at **8:00** with a maximum mood level of **100**.

During his day, several lectures will take place. As Gezi Wang attends these lectures, his mood level will drop.

> **For the first 60 minutes of a lecture, his mood level drops by 0.4/minute.**
> **After 60 minutes, his mood level drops by 0.8/minute.**

If his mood drops to or below 0, we will assume he has to be sent to the hospital.

When Gezi Wang does not have a lecture, he immediately returns to his dorm and sleeps.

> **His mood recovers by 0.5/minute in this situation.**
> His mood level will not exceed the maximum limit 100.

His day ends at **22:00**. Your goal is to find out his mood level at the end of the day.

Input description:
You will receive **an integer** in the first line, indicating the number of lectures today, as well as the number of remaining input lines.

In each of the following lines, you will receive **two 24-hour times**, separated by a hyphen (-). (**Sample: 11:00-13:00**) This indicates the time period of a lecture.

It's guaranteed that the inputs are valid, specifically that:

1. There is no lecture before 8:00 and after 22:00.
2. The lectures will be given in time order. In other words, after a line of "14:00-14:30", you won't receive a time period such as "9:00-10:00".
3. The end time of a lecture is always later than the starting time.
4. No lectures overlap, and there are always breaks between lectures. (And no matter how short a break is, he sleeps!)

Output description:
Your output will be one line at the beginning and one line in the end.

At the beginning, you should write:

"**How many lectures today?**", followed by a new line (\n).

After all inputs, you must give your output in the following format:

If Gezi Wang is not sent to hospital during his day, you should write:

"His mood level is * at the end of the day."** With a new line (\n).

<u>* The mood level should be printed with "printf" to display only 1 decimal point digit, even if it is an integer.</u>

Otherwise, you should write:

"Gezi Wang has been sent to hospital." Also with a new line (\n).

A correct program runs like this: (red indicates input and black indicates your output)

```
How many lectures today?
3
9:00-11:00
15:00-17:00
17:30-18:50
His mood level is 98.0 at the end of the day.
```

Problem 3. Floating Point Number Conversion (40 points)

We've learned number expression and bit operation in class. In this problem, we'll design a "new" floating-point number which is slightly different from the IEEE 754 standard, and implement the conversion between hexadecimal and our floating-point number.

Before the problem description, let's just recap on what we've learned so far.

(1) Number expression

The built-in types of C language like int, float, double…are actually bits (a binary sequence, e.g., 0b0100100…) in memory. For example, usually, char contains 8 bits, int and float contains 32 bits, and double contains 64 bits.

(2) Bit operations

We've learned bitwise operations and (&), or (|), not (~), xor (^) in class. The truth tables are shown in the following table.

a	b	a&b	a|b	a^b	~a
0	0	0	0	0	1
0	1	0	1	1	1
1	1	1	1	0	0
1	0	0	1	1	0

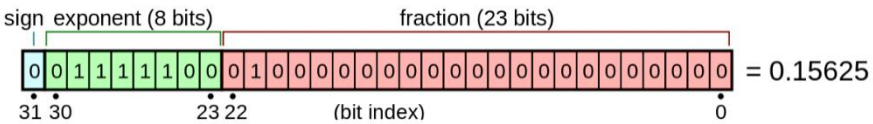
For example, assume A = 60 and B = 13 in binary format, they will be as follows.

```
 A = 0b00111100
 B = 0b00001101
A&B = 0b00001100
A|B = 0b00111101
A^B = 0b00110001
~A = 0b11000011
```

In this problem, you'll find these operations are useful when extracting or modify some bits in a number.

Now, let's design our floating-point number. First, we'll learn the floating-point encoding in our computer (IEEE 754 standard, https://en.wikipedia.org/wiki/IEEE_754).

A single-precision floating-point number is usually a 32-bit field in memory, it can be divided into 3 parts: sign, exponent and fraction. A standard float usually contains 23 fraction bits (0-22), 8 exponent bits (23-30) and 1 sign bit (31). For example, the bits 0x3e200000 (in binary, 0b00111110001000000000000000000000) is divided as:



The formula of converting a floating-point expression to a decimal value (V) is:

$$V = (-1)^{\text{sign}} \times (1 + 0.\text{fraction}) \times 2^{\text{exponent}-E}$$

Where E is:

$$\text{E} = 2^{\text{width of exponent field - 1}} - 1$$

For example, let's convert the above number to decimal.
Notice that the width of exponent field is 8 bits, so we can calculate E first:

$$E = 2^{8-1} - 1 = 127$$

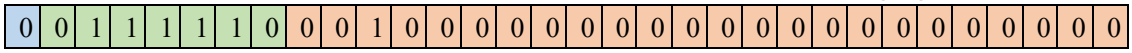
And the fraction part is $01000000000000000000000_{binary}$, so "0.fraction" is 0.01_{binary}. The expression of rational number in binary is similar to which in decimal. A digit to the left is twice as big as a digit to the right. (e.g., $0.1_{binary} = 0.5_{decimal}$, $0.01_{binary} = 0.25_{decimal}$).

Hence, $(1 + 0.fraction) = 1.01_{binary} = 1.25_{decimal}$. Finally, the value is

$$V = (-1)^0 \times 1.25 \times 2^{(124-127)} = 0.15625.$$

For more details about standard floats, you can refer to this link:
https://en.wikipedia.org/wiki/Single-precision_floating-point_format

But in this problem, we did a slight modification on the standard floating-point number. Our float still holds 32 bits in memory, and the width of sign field is still 1. **But the width of exponent is 7 bits, and width of fraction is 32-1-7=24 bits**, which is shown in the following figure



In this expression, the same bits above can be converted to:

$$V = (-1)^0 \times (1 + 0.001_{binary}) \times 2^{(62-63)} = (-1)^0 \times 1.125 \times 2^{(62-63)} = 0.5625$$

Input description:
You will receive a non-zero hexadecimal number only. It is guaranteed that, after converting into the floating-point number for this problem, its absolute value is greater than 0.00001.

Output description:
You need to convert it into a floating-point number using the rules given before, then print the converted number to the screen by this format:

```
printf("The number converts to %f\n", result);
```

A correct program runs like this: (red indicates input and black indicates your output)

```
0x3e200000
The number converts to 0.562500
```

```
0x3e600000
The number converts to 0.687500
```