Task 1

Your task is to create bit calculator. Function arguments are two bit representation of numbers `('101', '1', '10'...)`, and you must return their sum in decimal representation.

Requirements

- You are not allowed to use `parseInt` and `toString` methods

Examples

```
console.assert(calculate('10', '10') === 4);
console.assert(calculate('10', '0') === 2);
console.assert(calculate('101', '10') === 7);
```

Task 2

Write calculations using functions and get the results.

Requirements

- There must be a function for each number from 0 ('zero') to 9 ('nine')
- There must be a function for each of the following mathematical operations:
  - plus
  - minus
  - times
  - dividedBy
- Each calculation consist of exactly one operation and two numbers
- The most outer function represents the left operand, the most inner function represents the right operand

Examples

```
console.assert(seven(times(five())) === 35);
console.assert(four(plus(nine())) === 13);
console.assert(eight(minus(three())) === 5);
console.assert(six(dividedBy(two())) === 3);
```

Task 3

Write a function `defaultArguments`. It takes a function as an argument, along with an object containing default values for that function's arguments, and returns another function which defaults to the right values.

Requirements

- You cannot assume that the function's arguments have any particular names.
- You should be able to call `defaultArguments` repeatedly to change the defaults.

Examples

```
function add(a, b) {
  return a + b;
};

var add_ = defaultArguments(add, { b: 9 });
add_(10); // returns 19
add_(10, 7); // returns 17
add_(); // returns NaN

add_ = defaultArguments(add_, { b: 3, a: 2 });
add_(10); // returns 13 now
add_(); // returns 5

add_ = defaultArguments(add_, { c: 3 }); // doesn't do anything, since c isn't an argu
add_(10); // returns NaN
add_(10, 10); // returns 20
```

Task 4

The businessmen among you will know that it's often not easy to find an appointment. In this task we want to find such an appointment automatically. You will be given the calendars of our businessmen and a duration for the meeting. Your task is to find the earliest time, when every businessman is free for at least

that duration.

Requirements

- All times in the calendars will be given in 24h format "hh:mm", the result must also be in that format
- A meeting is represented by its start time (inclusively) and end time (exclusively) -> if a meeting takes place from 09:00 - 11:00, the next possible start time would be 11:00
- The businessmen work from 09:00 (inclusively) - 19:00 (exclusively), the appointment must start and end within that range
- If the meeting does not fit into the schedules, return `null`
- The duration of the meeting will be provided as an integer in minutes

Following these rules and looking at the example below the earliest time for a 60 minutes meeting would be 12:15.

Example Schedule

Person	Meetings
A	09:00 - 11:30, 13:30 - 16:00, 16:00 - 17:30, 17:45 - 19:00
B	09:15 - 12:00, 14:00 - 16:30, 17:00 - 17:30
C	11:30 - 12:15, 15:00 - 16:30, 17:45 - 19:00

Data Format

The schedule will be provided as 3-dimensional array. The schedule above would be encoded this way:

```
let schedules = [
  [['09:00', '11:30'], ['13:30', '16:00'], ['16:00', '17:30'], ['17:45', '19:00']],
  [['09:15', '12:00'], ['14:00', '16:30'], ['17:00', '17:30']],
  [['11:30', '12:15'], ['15:00', '16:30'], ['17:45', '19:00']]
];
```