Hong Kong Olympiad in Informatics 2014/15 Team Formation Test

Time allowed: 5 hours

Task Overview

Task	CPU time limit	Score
Conveyor Belt Sushi	1 second	$9 + 15 + 17 + 12 + 47 = 100$
Elevator	1 second	$12 + 21 + 42 + 25 = 100$
Congressman Lee Sin	1 second	$20 + 12 + 14 + 17 + 37 = 100$
Barcode Scanner	5 seconds	$20 + 10 + 10 + 10 + 10 + 10 + 10 + 10 + 10 = 100$

Notice:

Unless otherwise specified, inputs and outputs shall follow the format below:
- One space between a number and another number or character in the same line.
- No space between characters in the same line.
- Each string shall be placed in its own separate line.
- No trailing space(s) in each line.
- No empty lines, except that the input and output should end with the endline character.

C++ programmers should be aware that using C++ streams (cin/cout) may lead to I/O bottlenecks and substantially slower performance.

C/C++ programmers should use `"%lld"` for 64-bit integers I/O.

Some test cases are grouped. You may need to pass all test cases in that group to get points.

Conveyor Belt Sushi

Problem

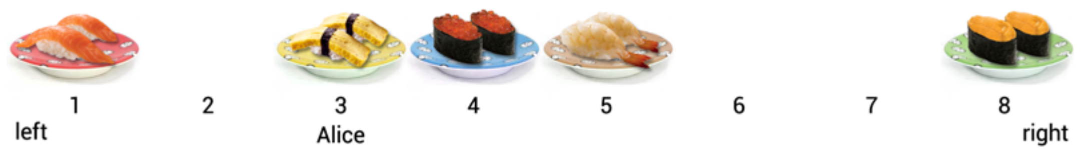
Alice is visiting a sushi restaurant which serves sushi with a very long, circular conveyor belt. When Alice arrives, the conveyor belt, which consists of N slots, is full of sushi. Initially, the sushi at slot i ($1 \leq i \leq N$) has label i. Alice has decided to sit at slot K ($1 \leq K \leq N$).

Each second, Alice would perform **one** of the below actions on the belt:

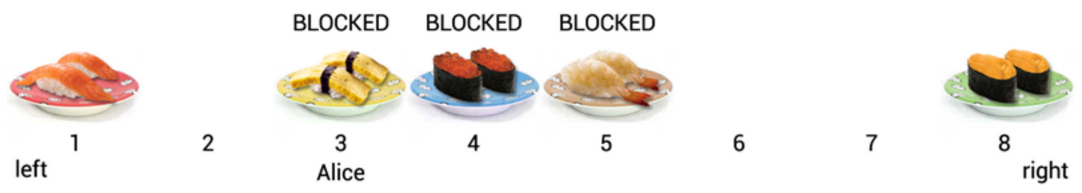
- **TAKE**: Take the sushi at the current slot K, leaving the slot empty. If the slot is currently empty, Alice takes nothing instead.
- **BLOCK**: Use her hands to block the sushi at the current slot K. The sushi at the slot, if not empty, will be marked as **BLOCKED**. Also, unblocked sushi that are immediately right of a **BLOCKED** sushi will be marked as **BLOCKED** also.
- **NOTHING**: Alice is enjoying the green tea / sushi and does nothing with the conveyor belt

After the action, all **unblocked** sushis on the belt moves one unit to the left, then all blocked sushis will become unblocked again. Notice that as the belt is circular, the left of the Slot 1 is Slot N.

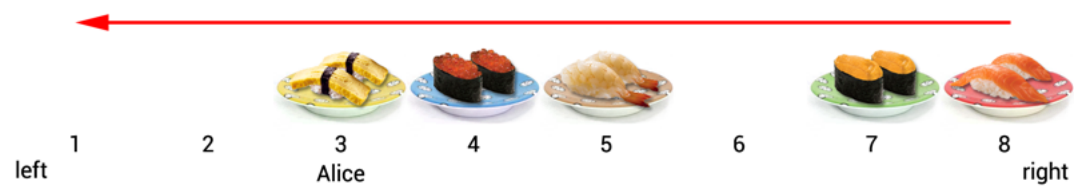
In this example, Alice sits at Slot $K = 3$. Consider that at some moment, the conveyor belt looks like the following picture:



If Alice performs the BLOCK action, then the sushi at Slot 3 will be marked with **BLOCKED**. Sushi at Slot 4 and 5 will be **BLOCKED** also.



Then, after 1 second, the conveyor belt looks like the follows, with the **BLOCKED** sushis not moving:



Alice is interested in the labels of the sushis she took for each **TAKE** operation. Can you help Alice?

Input

The first line of the input contains three integers N, K and Q, which are the total number of sushi disks, position of Alice, and number of seconds that Alice takes actions, respectively.

Next line contains Q characters each presenting one action, the characters will be always `T`, `B` or `N`, representing **TAKE**, **BLOCK** and **NOTHING**, respectively. It is guaranteed that there is at least 1 `T` action in the input.

Output

For each `T` action, print the label of the sushi that is taken from this action, or output `-1` if no sushi is taken.

Sample test

Input	Output
3 1 3 TBT	1 2

Input	Output
8 3 15 NNNTTNNTBTTTTTT	6 7 2 3 4 5 -1 8 1

Subtasks

Subtask 1 (9 points)
$1 \leq N \leq 1000$, $1 \leq Q \leq N$

Subtask 2 (15 points)
$1 \leq N \leq 1000$, $1 \leq Q \leq 1000$

Subtask 3 (17 points)
$1 \leq N \leq 100000$, $1 \leq Q \leq 100000$
In addition, all actions are either `T` or `N`.

Subtask 4 (12 points)
$1 \leq N \leq 100000$, $1 \leq Q \leq N$

Subtask 5 (47 points)
$1 \leq N \leq 100000$, $1 \leq Q \leq 100000$

In all test cases, $1 \leq K \leq N$.

Elevator

Problem

In Jones Enterprise, the tallest building in the world, there are M people waiting for the only elevator in order to get to a higher floor. More precisely, the building has H floors, and there are N floors with at least one people queuing. At the f_i^{th} floor, there are c_i people queuing, while the 1^{st} person being the foremost person and the c_i^{th} person being the rearmost person in queue. ($M = \sum_{i=1}^{N} c_i$)

As the elevator was engineered by Dr Jones, its structure is very different from ordinary elevators. Its interior is so narrow that people could only enter, line up or exit one by one. The elevator was made so long that its capacity is never a problem. If the elevator is empty, and A, B, C, D and E are queuing (A is the foremost person), after they have entered the elevator, they would end up in the order of E, D, C, B and A (E is closest to the door). As you can see, it is a terrible design when people want to exit:

When the elevator arrives at one's destination floor, all people in front of him/her are blocking his/her way. Therefore, those people should exit the elevator and push the waiting queue backward (they would occupy the front of the queue temporarily), letting him/her to leave the elevator. Afterwards, people in the waiting queue would enter the elevator one by one. For example, if C wants to exit, while F is waiting in the queue outside. Firstly, E and D would leave the elevator and push F backward so that C could get out. After that, D, E and F would enter the empty elevator one by one. After that, F would become the person closest to the door. Note that the elevator could only go upward, therefore once it arrives at a specific floor, it must let all people with that destination to leave.

As you are a better engineer than Dr Jones, you want to demonstrate the inefficiency of this design. As a result, you installed a sensor at the elevator door which counts the frequency of people passing through. More precisely, the sensor would maintain a counter which increases by one whenever a person enters or exits the elevator. Worrying about his reputation, Dr Jones removed your sensor in order to halt your investigation. As the old saying goes, when there is a will, there is a way. Given that the elevator is empty and waiting at 1^{st} floor initially, and the destination of each person waiting at different floors, could you calculate the reading of the sensor after all people have arrived at their destination?

Input

The first line consists of integers N and H.

In each of the next N lines, the first two integers are f_i and c_i, followed by c_i integers $d_{i,1}, d_{i,2}, ..., d_{i,c_i}$, representing that there are c_i ($1 \leq c_i \leq M - N + 1$) people waiting at f_i^{th} floor and the j^{th} person's ($1 \leq j \leq c_i$) destination is the $d_{i,j}^{th}$ floor ($f_i < d_{i,j} \leq H$).

Output

Output a single integer, the reading of the sensor after all people have arrived at their destination.

Sample test

Input	Output

2 6
1 2 2 5
3 3 4 4 6
	16

Input	Output

3 7
1 5 3 2 4 3 2
3 2 5 4
4 1 5
	24

Explanation of Sample 1

Floor	Queue (Door)	(Door) Elevator	Count
1	5 2		0
1		5 2	2
2	5	2	3
2	5		4
2		5	5
3	6 4 4	5	5
3		6 4 4 5	8
4	6	4 4 5	9
4	6	5	11
4		6 5	12
5		6	15
6			16

Subtasks

Subtask 1 (12 points)
$1 \le N \le M \le 3000$, $2 \le H \le 3000$

Subtask 2 (21 points)
$1 \le N \le M \le 3000$, $2 \le H \le 10^9$

Subtask 3 (42 points)
$1 \le N \le M \le 10^5$, $2 \le H \le 10^5$

Subtask 4 (25 points)
$1 \le N \le M \le 10^5$, $2 \le H \le 10^9$

In all test cases, $1 \le f_1 < f_2 < \ldots < f_N < H$

Congressman Lee Sin

Problem

Do you still remember the evil queen who imprisoned Snow White? After Prince Lee Sin defeated the evil army and saved Snow White, the evil queen was exiled. Also, Lee Sin has been elected to be a congressman in Snow White's kingdom.

However, the evil queen has never given up her ambition. She is planning to launch an attack to the Snow White's kingdom. There are N cities and $N-1$ roads. Road i connects two cities a_i, b_i and its length is l_i. One can reach all cities by roads no matter which city he starts from.

The queen is going to destroy zero or more roads. After the destruction, the kingdom will be divided into several disconnected parts. The queen wants every disconnected part to contain at most C cities. She wants to achieve it while keeping the sum of lengths of destroyed roads to be minimum.

Congressman Lee Sin has heard about the queen's evil plan. He would like to calculate the minimum sum of lengths of destroyed roads in order to anticipate the queen's attacks. Help congressman Lee Sin save the world!

Input

The first line contains 2 integers N and C.

$N-1$ lines follow. Each line contains 3 integers a_i, b_i and l_i.

Output

Output the answer in a single line.

Sample test

Input	Output
6 3	3
1 2 1	
2 3 1	
2 4 5	
4 5 1	
4 6 1	

Input	Output
9 4	4
1 2 1	
2 3 9	
3 4 6	
4 5 2	
5 6 3	
6 7 4	
7 8 7	
8 9 8	

Explanation

For sample 1, one possible solution is to destroy roads 1, 2 and 4.
For sample 2, one possible solution is to destroy roads 1 and 5.

Subtasks

Subtask 1 (20 points)
$N \leq 12$

Subtask 2 (12 points)
Any city can reach any other city by travelling at most 2 roads.

Subtask 3 (14 points)
$N \geq 2$
Exactly 2 cities each has exactly 1 neighbor.

Subtask 4 (17 points)
$C \leq 2$

Subtask 5 (37 points)
No additional constraint

In all test cases, $1 \leq N, C \leq 100$, $1 \leq a_i, b_i \leq N$, $1 \leq l_i \leq 10^7$

Barcode Scanner

Problem

You are about to implement a barcode scanner! Your task is to decode an image with an EAN-13 barcode in it **(except the first digit)**. Please refer to the Wikipedia article attached on how EAN-13 barcodes are encoded.

The barcode is embedded in a 300 pixels (H) $\times$ 400 pixels (W) 8-bit grayscale image. It is guaranteed that the barcode is decodable (tested by using a barcode scanner app). By referring to the samples given, design an algorithm to recognize the barcode and decode it.

Scoring

Subtask 1 is worth 20 points while Subtasks 2 to 9 are 10 points each. In a subtask, the characteristics of the images to be decoded would be similar to the samples given (see "Samples").

There are 2 tests in each of the 9 subtasks. The score for a test is determined by the length of the Longest Common Subsequence (LCS) of your output and the answer. The score for a subtask would be the lower score out of the two tests.

Length of LCS	0-3	4	5	6	7	8	9	10	11	12
Score (%)	0	10	20	30	40	50	60	70	80	100

For example, if the expected answer is `123456789012` but your program outputs `125690333312`, the LCS is `12569012` (length = 8) and you will get 50% score for the test.

Input

The input consists of 300 lines. Each line contains 400 integers. Those values are the pixel intensities of the image, which ranges from 0 (black) to 255 (white).

Output

Output the 2^{nd} to 13^{th} digit of the barcode in one line. Do not output the first digit.

Subtasks

Subtask 1 (20 points)
Same position, size, orientation, etc. Only the barcode value is different.

Subtask 2 (10 points)
The same filters are applied. Only the barcode value is different.

Subtask 3 (10 points)
The same filters are applied. Only the barcode value is different.

Subtask 4 (10 points)
Similar transformations are applied.

Subtask 5 (10 points)
Similar transformations are applied.

Subtask 6 (10 points)
Similar transformations and modifications are applied.

Subtask 7 (10 points)
Real image of similar style.

Subtask 8 (10 points)
No guarantees except that it can be decoded by barcode scanner app easily.

Subtask 9 (10 points)
No guarantees except that it can be decoded by barcode scanner app easily.

Samples

For your convenience, you can submit your program to "Barcode Scanner (samples)" (Not "Barcode Scanner") to test the samples.

You are recommended to inspect the following images in "Barcode Scanner (samples)" of contest system for higher resolution.

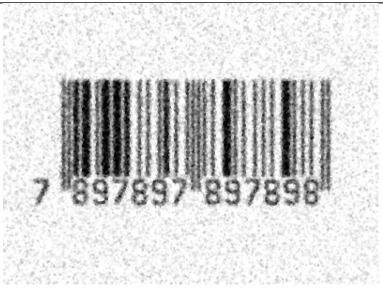
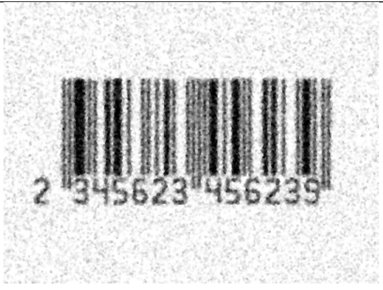
Subtask 1

1a.in	1b.in
1 234567 890128	9 876543 210982
1a.out: `234567890128`	1b.out: `876543210982`

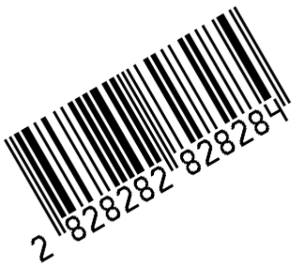
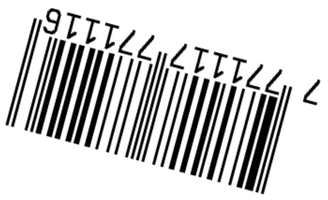
Subtask 2

2a.in	2b.in
1 928374 655642	1 357924 680139
2a.out: `928374655642`	2b.out: `357924680139`

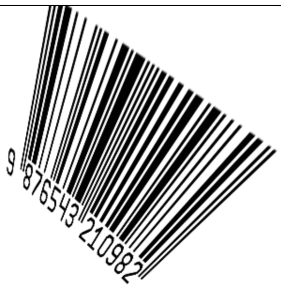
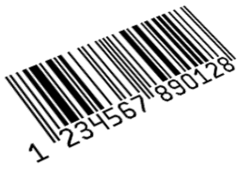
Subtask 3

3a.in	3b.in
	
3a.out: 897897897898	3b.out: 345623456239

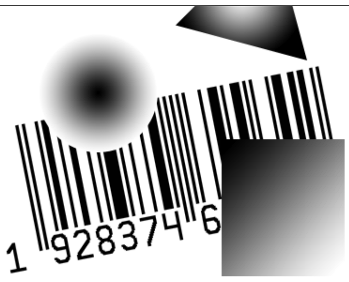
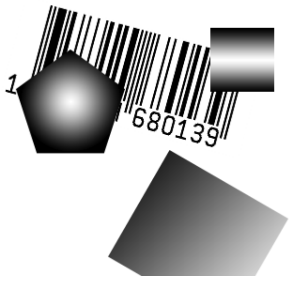
Subtask 4

4a.in	4b.in
	
4a.out: 828282828284	4b.out: 771117771116

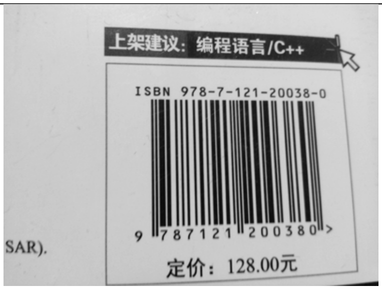
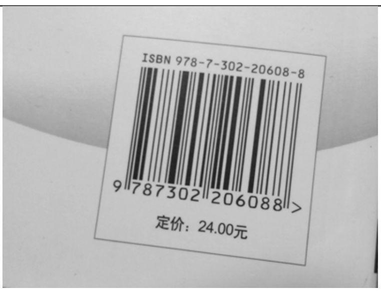
Subtask 5

5a.in	5b.in
	
5a.out: 876543210982	5b.out: 234567890128

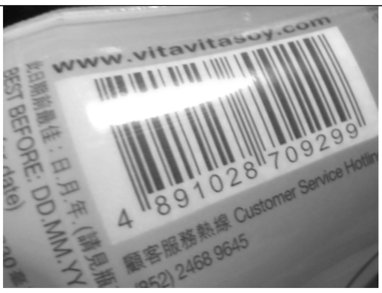
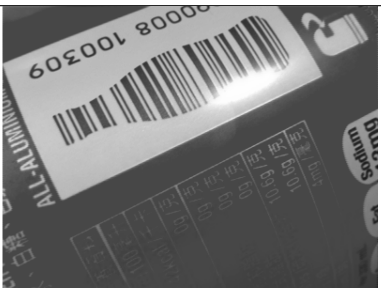
Subtask 6

6a.in	6b.in
	
6a.out: 928374655642	6b.out: 357924680139

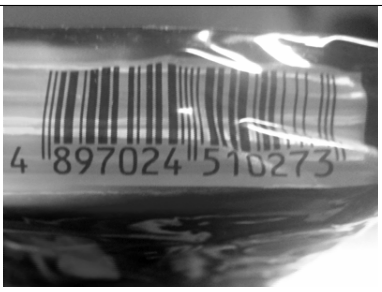
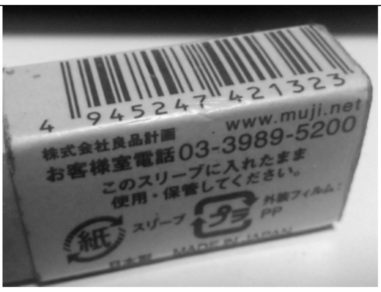
Subtask 7

7a.in	7b.in
	
7a.out: 787121200380	7b.out: 787302206088

Subtask 8

8a.in	8b.in
	
8a.out: 891028709299	8b.out: 890008100309

Subtask 9

9a.in	9b.in
	
9a.out: 897024510273	9b.out: 945247421323

International Article Number (EAN)

An **EAN-13** barcode (originally **European Article Number**, but now renamed **International Article Number** even though the abbreviation **EAN** has been retained) is a 13 digit (12 data and 1 check) barcoding standard which is a superset of the original 12-digit Universal Product Code (UPC) system developed in 1970 by George J. Laurer. The EAN-13 barcode is defined by the standards organization GS1.

5 901234 123457 >

GTIN-13 number encoded in EAN-13 barcode. The first digit is always placed outside the symbol.

The 13 digits in the EAN-13 barcode are grouped as follows:

- The left group: Digits 2-7. The left group also encodes digit 1, through a scheme of odd and even parity.
- The right group: Digits 8-13, digit 13 is the check digit.

Calculation of checksum digit

The checksum digit must be calculated from the data digits before it can be encoded. The checksum is calculated taking a varying weight value times the value of each number in the barcode to make a sum. The checksum digit is then the digit which must be added to this sum to get a number evenly divisible by 10 (i.e. the additive inverse of the sum, modulo 10).

Weight

The weight for a specific position in the EAN code is either 3 or 1, which alternate so that the final data digit has a weight of 3. In an EAN-13 code, the weight is 3 for even positions and 1 for odd position.

Weights for EAN-13 code:

Weights											
1	2	3	4	5	6	7	8	9	10	11	12
1	3	1	3	1	3	1	3	1	3	1	3

For example: 4006381333931

Calculation												
First 12 digits of the barcode	4	0	0	6	3	8	1	3	3	3	9	3
Weights	1	3	1	3	1	3	1	3	1	3	1	3
Multiplied by weight	4	0	0	18	3	24	1	9	3	9	9	9
Sum												89

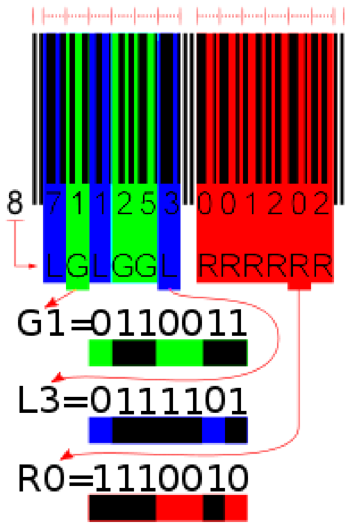
The nearest multiple of 10 that is equal or higher than the sum, is 90. Subtract them: 90 - 89 = 1, this is the last digit of the barcode.

Binary encoding of data digits into EAN-13 barcode

To encode an EAN-13 barcode, the digits are first split into 3 groups; the first digit, the first group of 6 and the last group of 6. The first group of six is encoded using a scheme whereby each digit has two possible encodings, one of which has even parity and one of which has odd parity. The first digit is encoded by selecting a pattern of choices between these two encodings for the next six digits, according to the table below. (Unlike the other digits, the first digit is not represented directly by a pattern of bars.) All digits in the last group of six digits are encoded using a single set of patterns which are the same patterns used for UPC.

If the first digit is zero, all digits in the first group of six are encoded using the patterns used for UPC, therefore, a UPC barcode is also an EAN-13 barcode with the first digit set to zero.

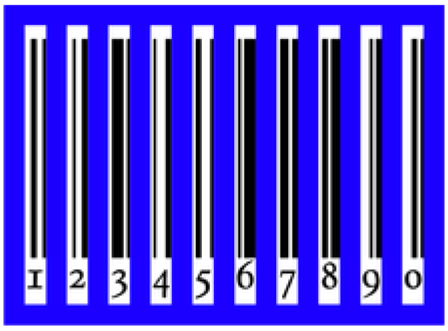
Structure of EAN-13		
First digit	**First group of 6 digits**	**Last group of 6 digits**
0	LLLLLL	RRRRRR
1	LLGLGG	RRRRRR
2	LLGGLG	RRRRRR
3	LLGGGL	RRRRRR
4	LGLLGG	RRRRRR
5	LGGLLG	RRRRRR
6	LGGGLL	RRRRRR
7	LGLGLG	RRRRRR
8	LGLGGL	RRRRRR
9	LGGLGL	RRRRRR



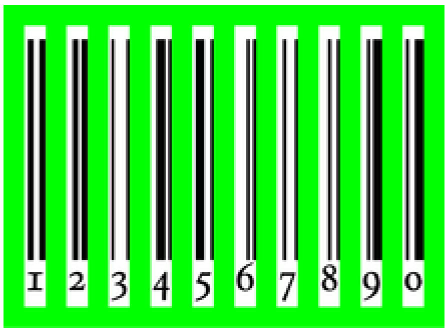
Encoding EAN-13

Note: You are not required to decode the first digit
Digit 2 is encoded using only L-codes and has an odd number of black bits (parity).
Digits 3 to 7 are encoded using only L-codes and G-codes
Digits 8 to 13 are encoded using only R-codes

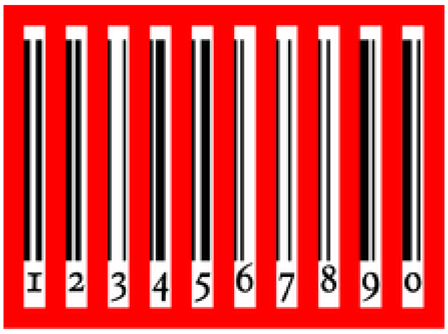
Encoding of the digits			
Digit	**L-code**	**G-code**	**R-code**
0	0001101	0100111	1110010
1	0011001	0110011	1100110
2	0010011	0011011	1101100
3	0111101	0100001	1000010
4	0100011	0011101	1011100
5	0110001	0111001	1001110
6	0101111	0000101	1010000
7	0111011	0010001	1000100
8	0110111	0001001	1001000
9	0001011	0010111	1110100



the numbers of code L



the numbers of code G



the numbers of code R

Note: Entries in the R-column are bitwise complements (logical operator: negation) of the respective entries in the L-column. Entries in the G-column are the entries in the R-column in reverse bit order. See pictures of all codes against a colored background.

How the GTIN (formerly EAN-13) is encoded

The code families UPC-A, EAN-8 and EAN-13 all use the same encoding. The encoded information is repeated in plain text below the barcode.

Coding

The barcode consists of 95 equally spaced areas. From left to right:

- 3 areas to encode the start marker.
- 6*7 = 42 areas making up the left group. This can be further subdivided into 6 subgroups, each consisting of seven areas. The subgroups encode digit 2-7. Each of these encodings can have even or odd parity. The parities taken together, encode digit 1.
- 5 areas to encode the marker for the center of the barcode.
- 6*7 = 42 areas making up the right group. This can be further subdivided into 6 subgroups, each consisting of seven areas. The subgroups encode digit 8-13. Digit 13 is the check digit. Digit 8-13 are all encoded with even parity.
- 3 areas to encode the end marker.

Each area can be black (meaning 1) or white (meaning 0). A maximum of four black areas can be grouped together, these make up a bar. Likewise a maximum of four white areas can be grouped together, these make up a space.

The start marker and the end marker are encoded as 101. The center marker is encoded as 01010.

Each digit in GTIN, except digit 1, consists of seven bits (seven areas). A decimal number between 0 and 9 is encoded so that it consists of two bars and two spaces. The combination of widths of the bars and spaces encodes the number.

The digits in the left group are encoded so that they always start with a space, and end with a bar. The digits in the right group are encoded so that they always start with a bar, and end with a space.

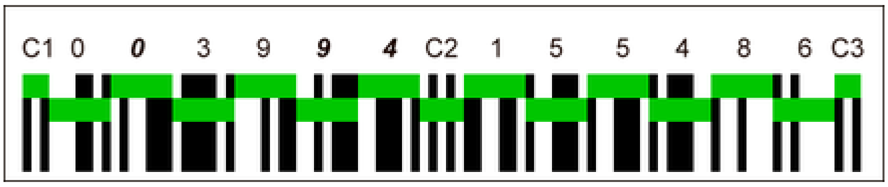
The encoding is described in the following table:

Digit	**Pattern**			**Bar width**		**Encoding of digit 1**
	left		**right**	**odd**	**even**	
	odd	**even**	**(even)**			
0	0001101	0100111	1110010	3211	1123	OOOOOO EEEEEE
1	0011001	0110011	1100110	2221	1222	OOEOEE EEEEEE
2	0010011	0011011	1101100	2122	2212	OOEEOE EEEEEE
3	0111101	0100001	1000010	1411	1141	OOEEEO EEEEEE
4	0100011	0011101	1011100	1132	2311	OEOOEE EEEEEE
5	0110001	0111001	1001110	1231	1321	OEEOOE EEEEEE
6	0101111	0000101	1010000	1114	4111	OEEEOO EEEEEE
7	0111011	0010001	1000100	1312	2131	OEOEOE EEEEEE
8	0110111	0001001	1001000	1213	3121	OEOEEO EEEEEE
9	0001011	0010111	1110100	3112	2113	OEEOEO EEEEEE

For each digit there are three similar encodings: *Left even* and *right* are mirror-symmetrical to each other. *Left odd* is the bitwise inverse of *right*.

The first digit from the left is always encoded with odd parity, and the last digit (on the right side) is always encoded with even parity. It thus does not matter whether the barcode is scanned from the left or from the right: The scanning software can determine what is the beginning and end of the barcode, with the help of the fact that the first digit should have odd parity and the last digit should have even parity.

Example

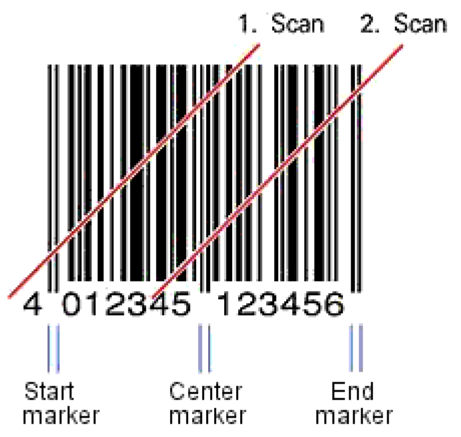


EAN-13 barcode. A green bar indicates the bars and spaces that encode a digit.

- C1, C3: Start-/Endmarker
- C2: Marker for the center of the barcode.
- 6 digits in the left group: 003994
- 6 digits in the right group, the last digit is the check digit: 155486
- A digit is encoded in seven bits, by two black bars and two white bars (space). Each bar can have a width between 1 and 4 bits.
- Parity for the digits from left to right: OEOOEE EEEEEE (O = Odd parity, E = Even parity)
- The first digit in the EAN code: The combination of parities of the digits in the left group encodes the digit 4.

The complete EAN-13 code is thus: 4 003994 155486.

Decoding



Scanning part of an EAN-13 barcode.

By using the barcode center marker, it is possible for a barcode scanner to scan just one half of the barcode at a time. This allows reconstruction of the code by means of a helical scan of the barcode by an angle of approximately 45 degrees.

Retrieved on 2015-04-21from http://en.wikipedia.org/wiki/International_Article_Number_%28EAN%29

Text is available under the Creative Commons Attribution-ShareAlike License.