Problems/Խնդիրներ

Ստորև ներկայացվում են խնդիրների պահանջները հայերեն և անգլերեն լեզուներով:

Problem 1/ Խնդիր 1

Completed Problem link version 1 - https://cutt.ly/SjXdXEe
Completed Problem link version 2 - https://cutt.ly/JjXd1dv

English

Suppose we have a 10X16 x + y axis. Please write a logic so that when inputing a value, we will get a visual representation of that value inside the axis. Example: The inputed value is 1.

```
**************************
*           11           *
*          111           *
*         1111           *
*        11 11           *
*       11  11           *
*      11   11           *
*     11    11           *
*           11           *
*           11           *
*           11           *
*           11           *
*           11           *
**************************
```

Հայերեն

Ունենք 10X16-ի վրա x+y առանցք: Անհրաժեշտ է գրել լոգիկա, որի մեջ ներմուծելով ինչ-որ արժեք, կստանանք նշված արժեքի վիզուալ տեսքը՝ առանցքի ներսում:
Օրինակ՝ տվյալ օրինակում ներմուծված արժեքը 1 է:

```
**************************
*           11           *
*          111           *
*         1111           *
*        11 11           *
*       11  11           *
*      11   11           *
*     11    11           *
*           11           *
*           11           *
*           11           *
*           11           *
*           11           *
**************************
```

Problem 2 / Խնդիր 2

Completed Problem link - https://cutt.ly/pjXd4hy

English

Your Problem is to determine whether the input string is valid, having the string containing only the characters '(', ')', '{', '}', '[' and ']'. Please note that input string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.

Note that an empty string is also considered valid.

Example 1:
Input: "()"
Output: true

Example 2:
Input: "()[]{}"
Output: true

Example 3:
Input: "(]"
Output: false

Example 4:
Input: "([)]"
Output: false

Example 5:
Input: "{[]}"
Output: true

Հայերեն

Ձեր առաջադրանքն է որոշել՝ արդյոք մուտքային տվյալները վավեր են՝ ունենալով միայն հետևյալ նիշերը '(', ')', '{', '}', '[' և ']':

Մուտքային տվյալները համարվում են վավեր, եթե:

1. Բաց փակագծերը փակվում են նույն տեսակի փակագծերով:
2. Բաց փակագծերը փակվում են ճիշտ հերթականությամբ:

Նշում. դատարկ մուտքային տվյալները նույնպես համարվում են թույլատրելի:

Օրինակները ներկայացված են վերևում:

Problem cash / Խնդիր cash

Completed Problem link https://cutt.ly/cjXd6G7

English

Let's imagine that the cashier owes the customer 41 cents change, and he has 25, 10, 5, and 1 cents for change. Guided by the "greedy" algorithm, the cashier will immediately want to give the maximum, in the first step. At this step, the optimal or best solution is to give out 25 pence. 41-25 = 16. That leaves 16 pence to be dispensed. Obviously, 25 pence is too

much, so that leaves 10. 16-10 = 6. Now we give out 5 pence according to the same principle, and then 1. Thus, the buyer will receive only four coins of 25, 10, 5 and 1 pence.

So, what is the minimum number of coins we need to give change?

- Given: 25, 10, 5, 1 cent coins.
- Ask the user how much change to give out.
- Calculate the minimum number of coins with which to do this.

Example 1:

O hi! How much change is owed?
Input: Change owed: 0.41
Output: 4

Example 2:

O hi! How much change is owed?
Input: Change owed: -0.41
How much change is owed?
Input: Change owed: -0.7
How much change is owed?
Input: Change owed: 0.55
Output: 3

Հայերեն

Եկեք պատկերացնենք, որ գանձապահը պետք է հաճախորդին 41 ցենտ մանր հանձնի, և նա ունի 25, 10, 5 և 1 ցենտ հանձնելու համար: Առաջնորդվելով «ագահ» ալգորիթմով ՝ գանձապահը առաջին իսկ քայլում կցանկանա անմիջապես տալ առավելագույնը: Այս քայլում օպտիմալ կամ լավագույն լուծումը 25 պենս տալն է , 45-25=16: Ակնհայտ է, որ 25 փենսը շատ է, ուստի թողնում է 10. 16-10 = 6. Այժմ մենք տալիս ենք 5 փենսի նույն սկզբունքի համաձայն, իսկ հետո ՝ 1: Գնորդը կստանա ընդամենը չորս մետաղադրամ 25, 10, 5 և 1 պենս.

Այսպիսով, ո՞րն է մետաղադրամների նվազագույն քանակը, որը մենք պետք է փոփոխություն կատարենք:

- Հաշվի առնելով, որ ունեք 25, 10, 5, 1 ցենտ մետաղադրամներ:
- Հարցրեք օգտվողին, թե որքան գումար պետք է տա փոփոխության համար:
- Հաշվեք մետաղադրամների նվազագույն քանակը, որով կարելի է դա անել:

Օրինակները ներկայացված են վերևում:

Problem credit / Խնդիր credit

Completed Problem link - https://cutt.ly/ijXfyDq

English

There are a lot of people with credit cards in this world, so those numbers are pretty long: American Express uses 15-digit numbers, MasterCard uses 16-digit numbers, and Visa uses 13- and 16-digit numbers.

All American Express numbers start with 34 or 37, most MasterCard numbers start with 51, 52, 53, 54, or 55 and all Visa numbers start with 4. But credit card numbers also have a “checksum” built into them, a mathematical relationship between at least one number and others.

Well, most cards use an algorithm invented by Hans Peter Luhn of IBM. According to Luhn’s algorithm, you can tell if the number is a credit card or not

Check this number, if it belongs to a credit card, print the name of that card, otherwise it is INVALID.

Example 1:
Input: Number: 4003600000000014
Output: VISA

Example 2:
Input: Number: 4003-6000-0000-0014

Input: Number: foo
Input: Number: 4003600000000014
Output: VISA

Example 3:
Input: Number: 6176292929
Output: INVALID

Հայերեն
Այս աշխարհում վարկային քարտեր ունեցող շատ մարդիկ կան, ուստի այդ թվերը բավականին երկար են: American Express-ը օգտագործում է 15-նիշանոց համարներ, MasterCard-ն օգտագործում է 16-նիշանոց համարներ, իսկ Visa - ն օգտագործում է 13 և 16-նիշանոց համարներ:
American Express-ի բոլոր համարները սկսվում են 34 կամ 37-ով, MasterCard-ի համարների մեծ մասը սկսվում է 51, 52, 53, 54 կամ 55-ից, իսկ Visa-ի բոլոր համարները սկսվում են 4-ից: Բայց վարկային քարտերի համարները ունեն նաև ներդրված "իսկիչ գումար", որը կառուցվել է նրանց, մաթեմատիկական կապը պահել, գոնե մեկ համարի և մյուսների միջև:

Քարտերի մեծ մասում օգտագործվում է ալգորիթմ, որը հորինել է Հանս Փիթեր Լուն IBM- ից: Ըստ Լուն - ի ալգորիթմի, դուք կարող եք իմանալ `համարը վարկային քարտ է, թե ոչ

Ստուգել տրված համարը , եթե պատկանում է վարկային քարտին տպել այդ քարտի անվանումը, հակառակ դեպքում INVALID:

Օրինակները ներկայացված են վերևում:

Problem readability / Խնդիր readability

Completed Problem link - *https://cutt.ly/TjXfcwd*

English
Implement a program that computes the approximate grade level needed to comprehend some text.

Reading Levels
E.B. White’s “Charlotte’s Web” book is between a second and fourth grade reading level, and Lois Lowry’s “The Giver” is between an eighth grade reading level and a twelfth grade reading level. What does it mean, though, for a book to be at a “fourth grade reading level”?
Well, in many cases, a human expert might read a book and make a decision on the grade for which they think the book is most appropriate. But you could also imagine an algorithm attempting to figure out what the reading level of a text is.

Harutyunyan

One such readability test is the Coleman-Liau index. The Coleman-Liau index of a text is designed to output what (U.S.) grade level is needed to understand the text. The formula is:

$index = 0.0588 * L - 0.296 * S - 15.8$

Here, L is the average number of letters per 100 words in the text, and S is the average number of sentences per 100 words in the text.

Let’s write a program called readability that takes a text and determines its reading level.

Example 1:

Input: Text: Congratulations! Today is your day. You're off to Great Places! You're off and away!

Output: Grade 3

The text the user inputted has 65 letters, 4 sentences, and 14 words. 65 letters per 14 words is an average of about 464.29 letters per 100 words. And 4 sentences per 14 words is an average of about 28.57 sentences per 100 words. Plugged into the Coleman-Liau formula, and rounded to the nearest whole number, we get an answer of 3: so this passage is at a third grade reading level.

Let’s try another one:

Example 2:

Input: Text: Harry Potter was a highly unusual boy in many ways. For one thing, he hated the summer holidays more than any other time of year. For another, he really wanted to do his homework, but was forced to do it in secret, in the dead of the night. And he also happened to be a wizard.

Output: Grade 5

This text has 214 letters, 4 sentences, and 56 words. That comes out to about 382.14 letters per 100 words, and 7.14 sentences per 100 words. Plugged into the Coleman-Liau formula, we get a fifth grade reading level.

As the average number of letters and words per sentence increases, the Coleman-Liau index gives the text a higher reading level. If you were to take this paragraph, for instance, which has longer words and sentences than either of the prior two examples, the formula would give the text an eleventh grade reading level.

Example 3:

Input: Text: As the average number of letters and words per sentence increases, the Coleman-Liau index gives the text a higher reading level. If you were to take this paragraph, for instance, which has longer words and sentences than either of the prior two examples, the formula would give the text an eleventh grade reading level.

Output: Grade 11

As the average number of letters and words per sentence increases, the Coleman-Liau index gives the text a higher reading level. If you were to take this paragraph, for instance, which has longer words and sentences than either of the prior two examples, the formula would give the text an eleventh grade reading level.

Հայերեն
Իրականացնել ծրագիր, որը հաշվարկում է գնահատման մոտավոր մակարդակը, որն անհրաժեշտ է որոշ տեքստ հասկանալու համար:

Է.Բ. Ուայթի «Charlotte's Web» գիրքը գտնվում է երկրորդ և չորրորդ դասարանների ընթերցանության մակարդակի միջև, իսկ Լոիս Լոուրի- ի «The Giver» գիրքը` ութերորդ դասարանի ընթերցանության մակարդակի և տասներկուերորդ դասարանի ընթերցանության մակարդակի միջև:
Ինչ է սա նշանակում "ընթերցանության չորրորդ մակարդակում" գտնվող գրքի համար:
Հաճախակի մասնագետ մարդը կարող է գիրք կարդալ եւ որոշում կայացնել դասի մասին,որի համար, իր կարծիքով, գիրքը առավել հարմար է: Բայց կարող եք պատկերացնել նաև, որ ալգորիթմը փորձում է հասկանալ, թե որն է տեքստի ընթերցման մակարդակը:

Շատ դեպքերում, մարդկային փորձագետը կարող է գիրք կարդալ և որոշում կայացնել այն դասարանի վերաբերյալ, որի կարծիքով, այդ գիրքն ամենահարմարն է: Բայց կարող եք պատկերացնել նաև, որ ալգորիթմը փորձում է հասկանալ, թե որն է տեքստի ընթերցման մակարդակը:
Ընթերցանության համար նման փորձերից մեկը Քոլման-Լիաու ինդեքսն է (index): Քոլման-Լիաու ինդեքսը նախատեսված է տեքստը հասկանալու համար անհրաժեշտ (ամերիկյան) գնահատականի մակարդակը ցուցադրելու համար: Բանաձև -
ինդեքս(index) = 0,0588 * L - 0,296 * S - 15,8
Այստեղ L- ը տեքստի 100 բառի համար տառերի միջին քանակն է, իսկ S- ը `նախադասության միջին քանակը տեքստի 100 բառի համար:
Եկեք գրենք ծրագիր, որը կոչվում է ընթեռնելիություն, որը վերցնում է տեքստ և որոշում դրա ընթերցման մակարդակը:

Example 1:
Օրինակը ներկայացված է վերևում:

Տեքստը, որը մուտքագրեց օգտվողը, ունի 65 տառ, 4 նախադասություն և 14 բառ: 14 բառի համար 65 տառ միջին հաշվով մոտ 464,29 տառ է 100 բառի համար: Իսկ 4 նախադասությունը 14 բառի համար միջինում կազմում է մոտ 28,57 նախադասություն 100 բառի համար: Մտնելով Քոլման-Լյաու բանաձևի մեջ և կլորացված մինչև մոտակա ամբողջ թիվը, մենք ստանում ենք 3-ի պատասխան. Այս հատվածը գտնվում է երրորդ դասարանի ընթերցանության մակարդակում:

Փորձենք ևս մեկը.

Example 2:
Օրինակը ներկայացված է վերևում:

Այս տեքստը բաղկացած է 214 տառից, 4 նախադասությունից և 56 բառից: Դա կազմում է մոտ 382,14 տառ `100 բառի համար, և 7,14 նախադասություն` 100 բառի համար: Միանալով Քոլման-Լիաու բանաձևին ` մենք ստանում ենք հինգերորդ դասարանի ընթերցանության մակարդակ:

Example 3:
Օրինակը ներկայացված է վերևում:

Նախադասության տառերի և բառերի միջին քանակի ավելացման հետ մեկտեղ Քոլման-Լիաու ինդեքսը տեքստի ընթերցման ավելի բարձր մակարդակ է տալիս: Եթե վերցնենք, օրինակ, այս պարբերությունը, որում բառերը և նախադասություններ ավելի երկար են, քան երկու նախորդ օրինակներից մեկում, ապա բանաձևը տասներկուերորդ դասարանում տեքստին կտա ավելի բարձր ընթերցանության մակարդակ:

Problem caesar / Խնդիր caesar

Completed Problem link - *https://cutt.ly/JjXf1lh*

plaintext	H	E	L	L	O
+ key	1	1	1	1	1
= ciphertext	I	F	M	M	P

English

Implement a program that encrypts messages using Caesar’s cipher.
Unencrypted text is generally called plaintext. Encrypted text is generally called ciphertext. And the secret used is called a key.
To be clear, then, here’s how encrypting HELLO with a key of 1 yields IFMMP:
More formally, Caesar’s algorithm (i.e., cipher) encrypts messages by “rotating” each letter by k positions. Denote the unencrypted text by the letter p, and pi - the letter in the text p, which is at position number i. Let's call the secret key the letter k, c is the ciphertext, and ci is the letter in the ciphertext that is at position i. Then we can calculate each letter of the cipher using the formula:

ci = (pi + k) % 26

Example 1:
Input: SimpleNumberKey: 1
Input: PlainText: HELLO
Output: CipherText: IFMMP

Here’s how the program might work if the user provides a key of 13 and a plaintext of hello, world:

Example 2:
Input: SimpleNumberKey: 13
Input: PlainText: hello, world
Output: CipherText: uryyb, jbeyq

Notice that neither the comma nor the space were “shifted” by the cipher. Only rotate alphabetical characters!

Example 3:
Input: SimpleNumberKey: 13
Input: PlainText: be sure to drink your Ovaltine
Output: CipherText: or fher gb qevax lbhe Binygvar

<u>Հայերեն</u>
Իրականացնել ծրագիր, որը գաղտնագրում է հաղորդագրությունները՝ օգտագործելով Կեսարի ծածկագիրը: Չծածկագրված տեքստը հիմնականում կոչվում է պարզ տեքստ: Գաղտնագրված տեքստը հիմնականում կոչվում է ծածկագրային տեքստ: Իսկ օգտագործված գաղտնիքը կոչվում է բանալի(key):
Որպեսզի պարզ լինի, ահա, թե ինչպես է HELLO- ի գաղտնագրումը 1 բանալով տալիս IFMMP:
Ավելի պաշտոնական, Կեսարի ալգորիթմը (այսինքն ՝ ծածկագիրը) գաղտնագրում է հաղորդագրությունները՝ յուրաքանչյուր տառը «պտտելով» ըստ k դիրքի: Նշեք չծածկագրված տեքստը p տառով, իսկ pi - տեքստում գտնվող p տառը, որը գտնվում է i դիրքում: Եկեք գաղտնի բանալին անվանենք k տառը, c- ը ծածկագրի տեքստն է, իսկ ci- ն ծածկագրի տեքստում գտնվող նամակն է, որը գտնվում է i դիրքում: Դրանից հետո, օգտագործելով բանաձևը, կարող եք հաշվարկել ծածկագրման յուրաքանչյուր տառ:

ci = (pi + k)% 26

Նկատեք, որ ոչ ստորակետը, ոչ էլ տարածությունը «չեն տեղափոխվել» ծածկագրով: Պտտեք միայն այբբենական նիշերը:
Օրինակները ներկայացված են վերևում:

Problem vigenere / Խնդիր vigenere

Completed Problem link - https://cutt.ly/XjXg3cd

English

The Vigenere cipher is somewhat safer than the Caesar cipher: it uses a word as the key and is difficult to break by hand. Each letter of the key generates a number, and as a result we get several keys to shift the letters.

If the number of letters in the message is not divisible by the key, we use only a part of the key in the last application of the key:

To find the value for the shift, we use the positions of each letter of our key **bacon** in the alphabet (a to z). We count from zero, like true programmers. And we shift each letter in the original text by a given number, as in Caesar's cipher, returning if necessary after z to the beginning of the alphabet. Thus, M will be shifted by 1, the first e will not be shifted at all, and the second e will be shifted by 2 positions. Below you see the original message, the painted key, and the result of its application.

```
Meet me at the park at eleven am
baco nb ac onb acon ba conbac on
Negh zf av huf pcfx bt gzrwep oz
```

Mathematics of the cipher:

ci = (pi + kj) % 26

Let p be some text, k be the keyword, kj be the jth letter of the key, pi be the letter number i in the original text, ci be the letter number i in the cipher. Then:

Example 1:
Input: SimpleTextKey: bacon
Input: PlainText: Hello
Output: CipherText: Ienzb

Example 2:
Input: SimpleTextKey: bacon
Input: PlainText: Meet me in the park at eleven am
Output: CipherText: Negh zf ip huf pcfx bt gzrwep oz

Example 3:
Input: SimpleTextKey: 25
Output: Keyword should be alphabetic keyword only.

Հայերեն

Վիժիներ ծածկագիրը որոշ չափով ավելի անվտանգ է, քան Կեսարի ծածկագիրը. Այն օգտագործում է բառը որպես բանալի և դժվար է ձեռքով կոտրել: Բանալու յուրաքանչյուր տառ առաջացնում է մի թիվ, և արդյունքում մենք ստանում ենք մի քանի բանալի, տառերը տեղափոխելու համար:

Եթե հաղորդագրության մեջ տառերի քանակը հավասարապես չի բաժանվում բանալու վրա, ապա բանալու վերջին օգտագործման ժամանակ մենք օգտագործում ենք դրա միայն մի մասը.

Որպեսզի գտնեք տեղափոխման արժեքը մենք օգտագործում ենք մեր becon բանալիի յուրաքանչյուր տառի այբբենական դիրքերը (a- ից z)։ Մենք հաշվում ենք զրոյից, ինչպես իսկական ծրագրավորողները։ Եվ յուրաքանչյուր տառը բուն տեքստի մեջ տեղափոխում ենք տրված համարով, ինչպես Կեսարի ծածկագրում, z- ից հետո անհրաժեշտության դեպքում վերադառնում ենք այբուբենի սկզբին։ Այսպիսով, M- ն կտեղափոխվի 1-ով, առաջին e- ն ընդհանրապես չի տեղափոխվի, իսկ երկրորդ e- ն կտեղափոխվի 2 դիրքով։ Ստորև տեսնում եք բնօրինակ հաղորդագրությունը, ներկված բանալին և դրա կիրառման արդյունքը։

```
Meet me at the park at eleven am
baco nb ac onb acon ba conbac on
Negh zf av huf pcfx bt gzrwep oz
```

Ծածկագրման մաթեմատիկան.

ci = (pi + kj) % 26

Թող p- ը լինի ինչ-որ տեքստ, k - բանալի բառ, kj - բանալու j- րդ տառը, pi - բնօրինակ տեքստի i տառի համարը, ci - տառը կոդավորման մեջ i համարի տակ գտնվող ծածկագիրը։

Օրինակը ներկայացված է վերևում։

<u>Problem substitution / Խնդիր substitution</u>

Completed Problem link - https://cutt.ly/1jXhiFL

<u>English</u>

In a substitution cipher, we “encrypt” a message by replacing every letter with another letter. To do so, we use a key: in this case, a mapping of each of the letters of the alphabet to the letter it should correspond to when we encrypt it. To “decrypt” the message, the receiver of the message would need to know the key, so that they can reverse the process: translating the encrypt text back into the original message .

A key, for example, might be the string NQXPOMAFTRHLZGECYJIUWSKDVB. This 26-character key means that A (the first letter of the alphabet) should be converted into N (the first character of the key), B (the second letter of the alphabet) should be converted into Q (the second character of the key), and so forth.

A message like HELLO, then, would be encrypted as FOLLE, replacing each of the letters according to the mapping determined by the key.

Here are a few examples of how the program might work. For example, if the user inputs a key of YTNSHKVEFXRBAUQZCLWDMIPGJO and a plaintext of HELLO.

Example 1:
Input: SimpleTextKey: YTNSHKVEFXRBAUQZCLWDMIPGJO
Input: PlainText: HELLO
Output: CipherText: EHBBQ

Example 2:
Input: SimpleTextKey: VCHPRZGJNTLSKFBDQWAXEUYMOI
Input: PlainText: hello, world
Output: CipherText: jrssb, ybwsp

Notice that neither the comma nor the space were substituted by the cipher. Only substitute alphabetical characters! Lowercase letters remain lowercase, and uppercase letters remain uppercase.
Whether the characters in the key itself are uppercase or lowercase doesn't matter. A key of VCHPRZGJNTLSKFBDQWAXEUYMOI is functionally identical to a key of vchprzgjntlskfbdqwaxeuymoi (as is, for that matter, VcHpRzGjNtLsKfBdQwAxEuYmOi).
And what if a user doesn't provide a valid key?

Example 3:
Input: SimpleTextKey: ABC
Output: Key must contain 26 characters.

Example 4:
Input: SimpleTextKey: 1 2 3
Output: Key must contain 26 characters.

Հայերեն

Փոխարինման ծածկագրում մենք «գաղտնագրում» ենք հաղորդագրություն ՝ յուրաքանչյուր տառ փոխարինելով մեկ այլ տառով: Դա անելու համար մենք օգտագործում ենք մի բանալի. այս դեպքում այբուբենի յուրաքանչյուր տառը համապատասխանեցնելով այն տառին, որը պետք է համապատասխանի այն կոդավորելիս: Հաղորդագրությունը «վերծանելու» համար ստացողը պետք է իմանա բանալին, որպեսզի նրանք կարողանան շրջել գործընթացը. Ծածկագրման տեքստը կրկին թարգմանելով բնօրինակ հաղորդագրության:

Բանալին, օրինակ, կարող է լինել NQXPOMAFTRHLZGECYJIUWSKDVB տողը: Այս 26 նիշանոց բանալին նշանակում է, որ A- ն (այբուբենի առաջին տառը) պետք է վերածվի N- ի (բանալու առաջին նիշը), B- ը (այբուբենի երկրորդ տառը) պետք է վերածվի Q- ի (երկրորդ նիշ) և այլն:

Ուստի HELLO- ի նման հաղորդագրությունը կոդավորված կլինի որպես FOLLE, յուրաքանչյուր տառերը փոխարինելով ըստ ստեղնով որոշված քարտեզագրման:

Ահա մի քանի օրինակ, թե ինչպես կարող է աշխատել ծրագիրը: Օրինակ, եթե օգտագործողը մուտքագրում է YTNSHKVEFXRBAUQZCLWDMIPGJO- ի բանալին և HELLO- ի պարզ տեքստ:

Example 1:
Օրինակը ներկայացված է վերևում:

Example 2:
Օրինակը ներկայացված է վերևում:

Նկատի ունեցեք, որ ոչ ստորակետը, ոչ էլ տարածությունը չեն փոխարինվել ծածկագրով: Փոխարինեք միայն այբբենական նիշերը: Փոքր տառերը մնում են փոքր, իսկ մեծերը՝ մեծ: Նշանակություն չունի բանալիում նիշերը մեծ են թե փոքր: VCHPRZGJNTLSKFBDQWAXEUYMOI ստեղնը ֆունկցիոնալորեն նույնական է vchprzgjntlskfbdqwaxeuymoi ստեղնին (ինչպես նաև VcHpRzGjNtLsKfBdQwAxEuYmOi):

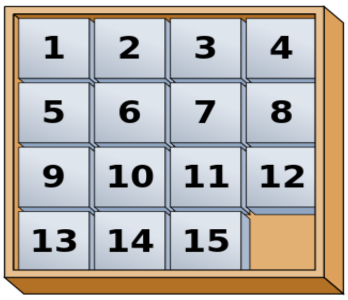
Եվ ի՞նչ կլինի, եթե օգտագործողը չի տալիս վավեր բանալի:

Example 3:
Օրինակը ներկայացված է վերևում:

Example 4:
Օրինակը ներկայացված է վերևում:

<u>Problem fifteen / Խնդիր fifteen</u>

Completed Problem link - *https://cutt.ly/ljXhl7P*



<u>English</u>

Write the game "fifteen"

Example 1:

Give the size game board fifteen, minimum size 3 and maximum size 9
Input: Size: 3

Output: 8|7|6
5|4|3
2|1|_
Input: Tile to move:

Example 2

Give the size game board fifteen, minimum size 3 and maximum size 9
Input: Size: 9

Harutyunyan

```
Output:  80|79|78|77|76|75|74|73|72
         71|70|69|68|67|66|65|64|63
         62|61|60|59|58|57|56|55|54
         53|52|51|50|49|48|47|46|45
         44|43|42|41|40|39|38|37|36
         35|34|33|32|31|30|29|28|27
         26|25|24|23|22|21|20|19|18
         17|16|15|14|13|12|11|10| 9
          8| 7| 6| 5| 4| 3| 2| 1| _
Input:   Tile to move:
```

<u>Հայերեն</u>

Գրիր «տասնհինգ» խաղը

Օրինակները ներկայացված են վերևում: