PROBLEM OF THE MONTH

December, 2015

2015-2016

# MATHEMATICS

## 5 points:

You have 100 beads, all different. How many different necklaces of length 42 can you make using these beads? The clasp on the necklace can be ignored, in the sense that 1-2-clasp-3-4 is the same as 1-clasp-2-3-4.

## Hint:

## Answer: $\frac{100!}{84 \cdot 58!}$

## Solution:

Let us first count the number of necklaces assuming that position of clasp matters. Starting with the clasp, we will be adding beads one by one. There are 100 possibilities of choosing the first one, 99 to choose the second one, etc, all the way to 100-41=59 ways of picking the bead number 42. This means that there are total of $100 \cdot 99 \cdot 98 \cdot .... \cdot 59$ different sequences of beads of length 42. This can also be expressed as:

$$100 \cdot 99 \cdot 98 \cdot .... \cdot 59 = \frac{100!}{58!}$$

Now let us recall that the position of a clasp does not matter. It can be moved to 42 different positions , and it will still be the same necklace.

In addition, the necklace does not have positive and negative direction, so sequence 1-2-3-4 represents the same necklace as the one with the reverse order: 4-3-2-1. This doubles the number of sequences that represent the same neckless Therefore, the above result should be divided by number $42 \cdot 2 = 84$, to avoid this multiple counting. This gives:

$$\frac{100!}{84 \cdot 58!}$$

## 10 points:

Solve the following equation:

$$\underbrace{\cfrac{1}{1-\cfrac{1}{1-\cfrac{1}{1-\ddots-\cfrac{1}{1-\cfrac{1}{1-x}}}}}}_{2015\ fractions} = x$$

## Hint:

(i) Denote $f(x) = \frac{1}{1-x}$ and try to find a pattern using this notation.

(ii) Note, that an equation may, or may not have real solutions.

## Answer:

There are two correct answers to this problem.

(i) No real solutions.

(ii) Two complex solutions, $x_{1,2} = \frac{1}{2} \pm i\frac{\sqrt{3}}{2}$

## Solution:

Denote $f(x) = \frac{1}{1-x}$. Note that $f(f(x)) = 1 - \frac{1}{x} = f^{-1}(x)$, so $f(f(f(x))) = x$, and applying $f(x)$ $3n$ times (and 2016 times in particular) we always obtain $x$. Therefore, the above equation is equivalent to $f(x) = x$, or, $x^2 - x + 1 = 0$. This equation has no real solutions (note, that if

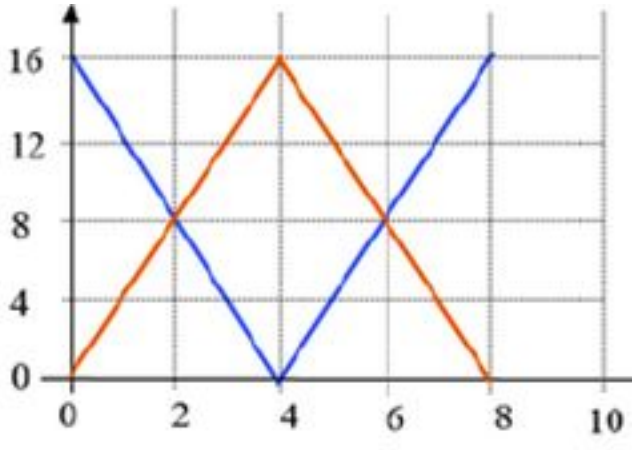
the function is applied 2016 times, then the equation is an identity, satisfied for any $x$). If you know complex numbers, you may write the two complex roots of this equation, although this is not required for the full score.

# PHYSICS

## 5 points:

Two cars (blue and red) move in the same direction. The velocity-time graphs below represent their motion (time is given in minutes and the velocity in m/s). At initial moment of time, the red car is 1440 m ahead of blue car. Find the maximum distance between two cars.

*Velocity, m/s*




*time, minutes*

**Hint**: the distance traveled is given by the area under the velocity graph.

**Answer**: 2400m

**Solution**: The change of the distance between red and blue cars is given by the area between red and blue curves (negative if red is below blue). Each "square" corresponds to 2min x 4m/s = 480m. During the first two minutes the distance decreased by two "squares". From 2 to 4 minutes it increased by 2 squares and then increased again by 2 squares between 4 and 6 minutes and started to decrease again. The maximal change is at time equal 6 minutes and is equal to 2 squares, that is 960m. The total distance between cars at that point is 1440+960=2400m.

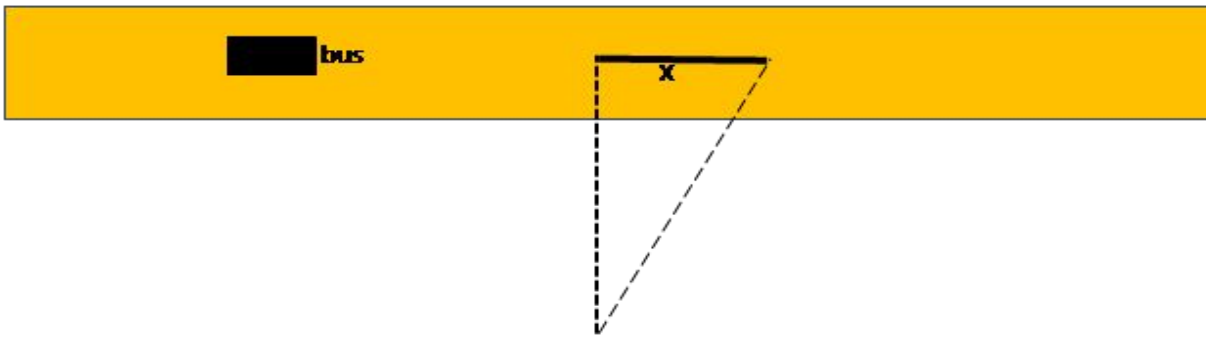
## 10 points:

A bus is moving along the straight road at a constant speed of 5 m/s. A boy, standing 30 meters away from the road, sees the bus when at the moment the bus is 50 meters away from him.
Is it possible for the boy to catch the bus, by reaching some point on the road before the bus reaches this point, if the boy can run at the maximum speed of 3 m/s? Explain.

**Hint**: try to find the point x on the road which the child can reach simultaneously with the bus.

**Answer:** yes, it is possible. This point is at x=22.5m from the point on the road closest to the initial position of the child.

**Solution:** Draw a perpendicular line from a current position of a child to the road (see diagram below).




child

This shortest distance will not work, since a child moves slower than the bus. A child should run in the direction slightly away from the bus. Let's call this advance distance x. Then the bus will travel the distance of (40+x) and the child would travel the distance of ($\sqrt{30^2 + x^2}$). Let's make the time for the bus and child equal to each other and solve for x:

$$\frac{40 + x}{5} = \frac{\sqrt{30^2 + x^2}}{3}$$

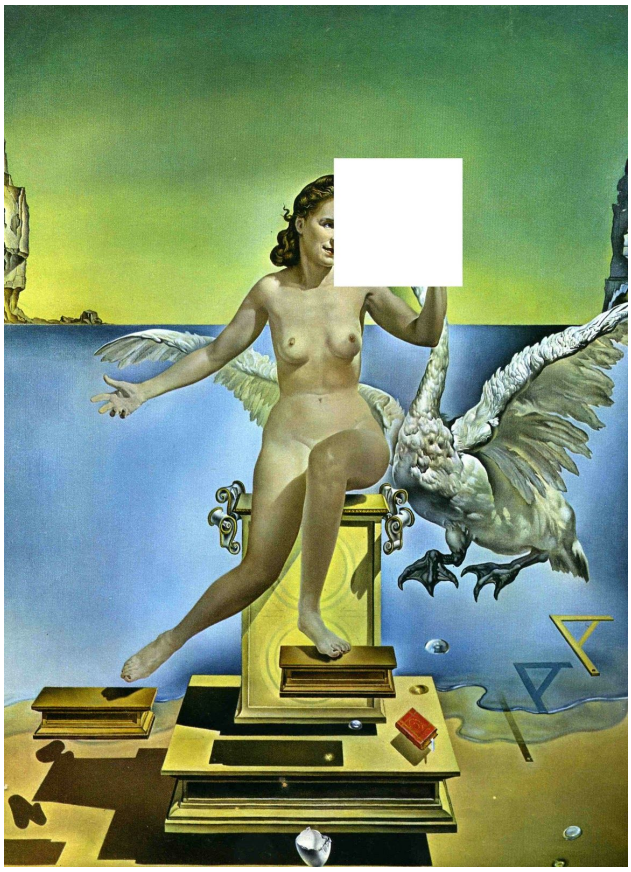
This equation has only one solution x=22.5 m, so it is possible for the child to reach a point on the road at the same moment as the bus. After 12.5 seconds both objects would meet: the bus would travel the distance of 62.5 m and the child would move along the hypotenuse the distance of 37.5 m.

# CHEMISTRY

## 5 points:

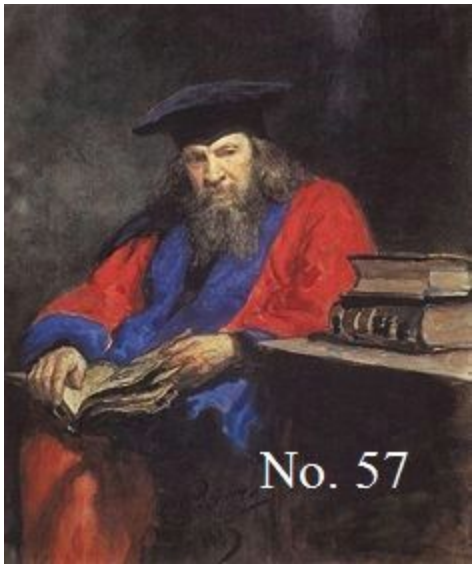
Imagine you are playing the Escape the Room game. To open the lock, you need to know some secret word (probably, four letters long). You tried almost all hints that you found in the room, but your efforts were futile. The last three objects that seem potentially useful are the following:

1. A fragment of the photocopy of some painting:



2. A glass beaker with hot water. There is a test tube in the beaker, and there are few drops of some liquid metal in the tube. The tube is sealed. When you take the tube out of the beaker and let it cool down, the metal solidifies. When you put the tube into the hot water, the metal melts again.

3. A portrait where some odd message is added:




These three objects are your last hope, because you have no other idea on how to obtain the secret word.
Try to find that code.
PS. And, yes, we all live in the Internet era, so the whole power of Google is available to you (you have your smartphone with you, and the coverage is excellent).

**Hint:** Try to find what is the name the man on the second portrait, and what was his major contribution to science.

## Answer:

The secret word is “Gala”.

## Solution:

The author of the first painting (“Leda Atomica”) is Salvador Dali, and the woman in this picture is Dali's wife (she can be found on many Dali's paintings). The name of Dali's wife was Elena Diakonova (she was Russian), but Dali called her Gala.
The second picture gives you a hint "La", and the metal in the tube has a melting temperature above 25 degrees and below 100 degrees Celsius (boiling temperature of water). Most likely it is gallium (Ga). Other metals exist with melting temperatures is this range (e.g. sodium, cesium, francium), but they belong to the group of alkaline metals. They are extremely dangerous, so no reasonable person will use them for the "Escape the room" game. In addition, francium is extremely expensive and radioactive, so it is physically impossible to use it for this game. Therefore, the only candidate is gallium, which is relatively nontoxic and safe metal (and even is used in high temperature thermometers instead of mercury).

To summarize, “Gala” can be deduced from the first picture, although that is not the only possible answer (another answer is, for example, “Leda”). However, two other hints, combined with the first live you the only possibility: “Gala”, “La” and “Ga” means only “Gala”.

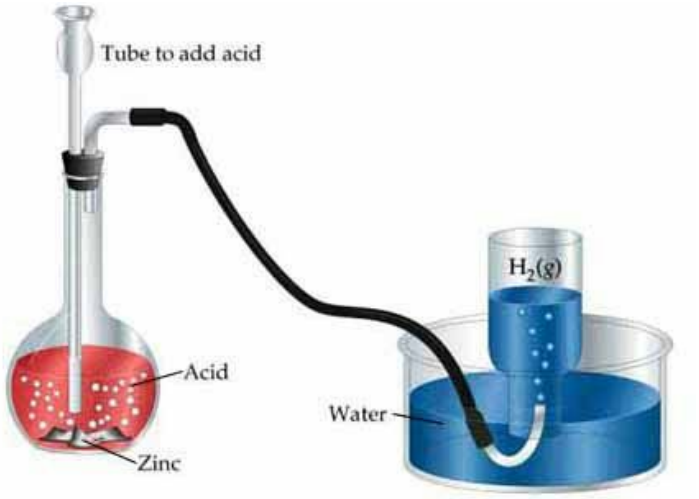
## 10 points:

When Bob, a Chemistry lab technician, came to Alice’s lab, she was staring at some piece of a strange rock at her table. “What’s that?” Bob asked. “Are you contemplating some new experiment?”

“Yes, Bob” Alice answered. “This rock is a fragment of *Chelyabinsk* meteorite. It exploded about a Russian city Chelyabinsk a couple years ago. My friend brought me a small fragment he picked up near the epicenter of the explosion, and I am going to take a part of this fragment to do its chemical analysis during the next class.”

“Are you sure we will be able to do a comprehensive chemical analysis using our equipment?” - Bob asked.

“Of course, no. But we can measure the iron content in it. Wikipedia says there is about 10% of elementary iron in *Chelyabinsk*, and I think we can try to check is that is correct. To determine the percentage of iron, you Bob will take a small piece of this rock (about 50 grams) and grind it using our mortar and pestle. After that, weigh the amount of the powder, put it into the Cavendish’s apparatus (shown on the picture below), and add hydrochloric acid




to the powder. Of course, in our case the meteorite powder plays the role of zinc: all hydrogen will be generated during the reaction of the meteorite iron with HCl.”

“Ok, I understand, Alice”, Bob says. “I add no zinc, and I use the meteorite material instead. What is my next step?”

“When the evolution of the hydrogen gas cease, measure the gas volume and calculate the iron content. Do you need more detailed instruction?”

“No, Alice, I understand. Alice, may I do this experiment with our students by myself? I believe I have enough experience, so your participation is not necessary.”

“Yes, Bob, do that,” Alice said. “Good luck. But don’t forget to take excess of HCl, otherwise your results will be misleading.”.

After the class, Bob came to Alice and said: “Alice, we took 53 grams of the meteorite material and 200 mL of 20% HCl. The volume of gas was 1.95 L.”
“Good job, Bob. And what does that mean? What is the iron content in the meteorite? Is Wikipedia right or wrong?”
“Alice, Wikipedia is ….”

What did Bob say?

**Hint:** Iron reacts with HCl according to the equation:

$Fe + 2\ HCl \rightarrow FeCl_2 + H_2$

That means one atom of iron produces exactly one molecule of the hydrogen gas.

## Answer:

Yes, Bob was right.

## Solution:

From the equation of this reaction:
$Fe + 2\ HCl \rightarrow FeCl_2 + H_2$
we conclude one iron atom produces exactly one molecule of hydrogen. That means, one *mole* of iron (i.e. 56 grams) yields 22.4 liters of the hydrogen gas. Bob obtained 1.95 L, which means the amount of iron in the sample was $56 \cdot 1.95/22.4 = 4.88$ grams. If 4.88 g of iron are in 53 g of the meteorite material, then 100 grams of the same material contain $(4.88/53) \cdot 100 = 9.19$ grams, or 9.19% of iron. That is what Wikipedia says (approximately 10%).

# BIOLOGY

## 5 points:

It is known that some animal species (wolves, geese, monkeys, etc) form communities with a very complex social structure, where a well developed hierarchy exists, and the members of the community establish strong personal relationships with each other. Obviously, to make that possible, the animals have to be capable of recognizing their congeners personally. Accordingly, in the communities of those animals who are incapable of recognizing each other personally (most fish species, majority of birds) no hierarchy or other forms of social structure exist (as a rule). Indeed, when the community is totally anonymous, it is almost impossible to propose the mechanism of formation of personal relationships. (*To demonstrate this idea, let's imagine some Internet community (for example, a chat) whose members are not allowed to have any nicknames, and who get a new random avatar every time they post a new message. Obviously, it would be very difficult for the members of such a chat to form any type of a more or less organized community.*)

Lizards, as well as other reptiles, are incapable of recognizing each other personally. However, in nature they form a community with a strong hierarchy: the strongest male occupies the best burrow in the most comfortable place, and it controls the greatest territory that it uses for hunting. The lizards establish the control over their territory by attacking invaders, so after a series of fights each member of the community gets their own hunting area, depending on their strength and fighting capabilities. Interestingly, the lizards form stable pairs, where the strongest male and the strongest female live together in their own burrow and use (and protect) the same territory. Despite that, even after a long period of living together the male and the female lizards do not recognize each other personally (a male lizard cannot recognize "his" female in the presence of other female lizards when they are placed in some artificial landscape).

*Can you explain the mechanism of formation of stable couples between lizards despite the fact that they (i) do not recognize each other personally, and (ii) have to attack invaders to protect their territory?*

## Answer:

The key fact here is that male lizards do not attack females and vise versa. However, every male attacks all other males, and every female attack all other females. As a result, we actually have *two* populations of lizards, each of which establish their own hierarchy and divide territory. However, since males and females are neutral towards each other (no aggression and no friendship), a male and a female shares the same burrow and the same hunting area. Obviously, since the male and female that share some territory mate with each other, not with other lizards. That creates an impression of a stable couple.

In summary:

- a male lizard protects “his” territory from other males and does not attack females;
- a female lizard protects “her” territory from other females does not attack males;

- as a result, “his” territory roughly coincides with “her” territory (a patch of land around the burrow they share), and all other lizards have no access to it. However, there is no love or friendship in this “union”: both members of this “family” are being held together only by their common home.

## 10 points:

The development of reliable anti-AIDS therapy is a formidable task. One of the features that makes it extremely difficult is the fact that the enzyme responsible for copying HIV’s genome (this enzyme is called “reverse transcriptase”) works very unreliably and makes a lot of errors. That means, majority of viral particles are mutants that have numerous amino acid replacements in every HIV protein, including the reverse transcriptase itself. Normally, the non mutated virus copies have an advantage over the mutants, but in a situation when HIV is subjected to some antiviral therapy, some mutant copies of HIV may have an advantage, so the new population of mutant virus forms that is not sensitive to this particular anti-HIV drug. In some cases, just a couple of weeks are needed for HIV to develop resistance to a particular drug.

HIV is a very primitive virus, and its genome encodes just few proteins. One of the key proteins is a reverse transcriptase itself, and this protein is the major target for anti-HIV therapy. Accordingly, most anti-HIV drugs (such as zidovudine) are mimics of nucleotides (which are the building blocks a reverse transcriptase uses to synthesize the viral DNA). Usually, HIV reverse transcriptase incorporates such a “pseudo-nucleotide” into the growing DNA chain, and that leads to termination of the viral DNA synthesis. Since the reverse transcriptase is a very imprecise enzyme, it utilizes the nucleotide analogs that are ignored by our own DNA synthesizing enzymes. That is why zidovudine and similar anti-HIV drugs are almost non-toxic to humans, but are capable of killing HIV.

However, since the reverse transcriptase makes a lot of errors, HIV, as well as the reverse transcriptase itself, evolve very rapidly, and new HIV mutants become more precise, and they ignore zidovudine and other first generation anti-HIV drugs. When the scientists faced this problem, they initially believed they are doomed to develop more and more novel anti-HIV drugs, each of which will be becoming obsolete rapidly.

However, about 10 years ago, the scientists developed a totally new strategy: they started to treat patients not with a single anti-HIV drug, but with a combination of four drugs: three of them were designed to suppress HIV reverse transcriptase, and the fourth one killed a second HIV enzyme. Surprisingly, the scientists found that HIV was incapable of developing resistance towards this type therapy, and no AIDS symptoms have been being developed in the patients who were taking this medication continuously and without interruptions.

*Can you tell why that therapy does not lead to development of resistance?*

## Answer:

To understand why this therapy works, one can understand who things: the mechanism of HIV drug-resistance, and the mechanism of the conventional anti-HIV therapy.

The main enzyme responsible for copying HIV genome works very fast, but it is doing its job very poorly: an overwhelming majority of copies of HIV genome have numerous errors, and as a rule, such particles are not viable. However, this variability provides a virus with a very serious advantage: when you start to treat the virus with some anti-HIV drug, you kill just a “normal” virus, however, a population of viral particles produces an enormous number of variants of HIV that are resistant towards all conceivable drugs.

Now let’s tell about a concrete mechanism of anti-HIV therapy. Majority of first generation drugs (the first representative of this class was zidovudine) are the mimics of normal *nucleotides* (the building blocks both the virus and the cell use to build their DNA). The synthesis proceeds step by step: cellular DNA polymerase or HIV reverse transcriptase crawls along the template DNA strand it is copying, and adds new nucleotides to complement the nucleotide in the template DNA molecule. After the new building block (a nucleotide) has been added, the polymerase (or HIV reverse transcriptase) moves one step forward, and is ready to incorporate the next nucleotide. The difference between zidovudine and normal nucleotides is that the former can be incorporated into the growing DNA strand, but after that further elongation of the DNA becomes impossible. However, fortunately for us, human enzymes that are responsible for copying our own DNA ignore zidovudine, and do *not* incorporate it into our own DNA. In contrast, HIV’s enzyme does incorporate in into its own DNA, and, as a result, viral replication is stopped.

That is how HIV therapy worked in 90s. However, as we already know, HIV is extremely variable, and its enzyme responsible for DNA synthesis evolved quickly. Now, modern HIV versions ignore zidovudine, because HIV reverse transcriptase does not recognize it any more as a “building block”.

The next step in the battle with HIV is to develop new drugs that work in the same manner, but that are still capable of tricking HIV’s reverse transcriptase. Scientists developed at least fifty drugs of that kind, but there is no guaranty HIV will not become resistant to them too.

Upon meditation, scientists asked themselves: “If HIV will become resistant towards all these drugs, does it mean its reverse transcriptase will work more precisely, and make less mistakes? Let’s check that.”

They started to treat HIV with several zidovudine type drugs simultaneously - and teh virus developed resistance to them all. However, the scientists observed that was achieved at cost of significantly increased fidelity of DNA synthesis by reverse transcriptase. In other words, the number of mutated viral particles dropped dramatically. When the scientists found that, they added *one more* drug that acted by the different mechanism - and they found HIV became incapable of developing resistance towards this fourth drug. That discovery laid the foundation for the modern anti-HIV therapy: three zidovudine type drugs force the HIV to become less variable, and the fourth drug kill the overwhelming majority of viral particles.

Thanks to this discovery, the patients with AIDS can live as long as any healthy human lives - provided that they continue to take this four component medicine without interruptions.

# COMPUTER SCIENCE

- You can write and compile your code here: http://www.tutorialspoint.com/codingground.htm
- Your program should be written in C, C++, Java, or Python
- Any input data specified in the problem should be supplied as user input, not hard-coded into the text of the program.
- Please make sure that the code compiles and runs on http://www.tutorialspoint.com/codingground.htm before submitting it.
- Submit the problem in a plain text file, such as .txt, .dat, etc. **No .pdf, .doc, .docx, etc!**

## 5 points:

One-Dimensional "XOR" Game of Life

The Game of Life (https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life) is a game developed by the British mathematician John Horton Conway in 1970. In this two-dimensional game the state of the next generation of cells placed in a rectangular grid fully depends on the state of the neighbor cells in the previous generation. In this assignment, you will develop a simplified, one-dimensional Game of Life. As an input you will be a given a string that consists of 0s and 1s. This string represents a colony of cells, with each digit in this string representing a single cell. Cell state could be either 0 or 1. The state of the cell in the next generation will be logical XOR operation of its neighbors' states in the current generation. What this means is that if the neighbors of the cell are equal, 1 and 1 or 0 and 0, the cell will turn into 0 in the next generation. However, if we neighbors of the cell are different, 0 and 1 or 1 and 0, the cell will turn into 1. (Note: border cells, i.e. the leftmost one and the rightmost one, have one of the neighbors outside of the colony, and the state of those "outside neighbors is always 0). So, our colony of cells will change generation after generation. To make the output more appealing, you will print each generation of the colony on a separate line, you will use space instead of 0 and "x" instead of 1. For example, given the input of 1101010, first 6 generations of the colony will look like this:

```
xx x x
xx    x
xxx  x
x xxx x
  x x
 x   x
```

Write a program that takes a string of 1s and 0s on input and prints out 16 generations of the colony. You can use the following input :

00000000000000000000000000000000000000000000000000000000010000000000000000000000000000000000000000000000000000000000000

and see what comes out.

## Solution:

**In Java:**

```
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class GameOfLife {
  public static void main(String[] args) throws Exception {
    int n = 16; // number of generations of the colony
    int[] a;    // last generation
    int[] b;    // new generation
    String str;
    boolean useRandom = true;

    if(!useRandom) {
      // input a colony string
      System.out.println("Enter an initial colony string of 0's and 1's:");
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      str = br.readLine().trim();
    }
    else {
      // generate random string
      int len = (int)(50*Math.random())+2; // random length from 2 to 51 characters
      StringBuilder s = new StringBuilder(len);
      for(int i=0; i<len; i++) {
        if(Math.random() < 0.5) // '0' and '1' are equally weighted (50/50)
          s.append('0');
        else
          s.append('1');
      }
      str = s.toString();
      System.out.println("random input string: ");
      System.out.println(str);
    }

    // verify input
    if(str.length() < 2)
      throw new Exception("input length must be >= 2");
    for(int i=0; i<str.length(); i++) {
      if(str.charAt(i)!='0' && str.charAt(i)!='1')
        throw new Exception("input must contain only '0' or '1'");
    }

    // allocate arrays
    a = new int[str.length()];
    b = new int[str.length()];

    // convert string to integer array
    for(int i=0; i<str.length(); i++) {
      if(str.charAt(i) == '0')
        a[i] = 0;
      else
        a[i] = 1;
```

```
    }

    // produce next generations
    for(int i=0; i<n; i++) {
      b[0] = 0 ^ a[1]; // left boundary
      for(int j=1; j<a.length-1; j++) {
        b[j] = a[j-1] ^ a[j+1];
      }
      b[a.length-1] = a[a.length-2] ^ 0; // right boundary

      // print
      for(int j=0; j<b.length; j++) {
        if(b[j] == 0)
          System.out.print(' ');
        else
          System.out.print('x');
      }
      System.out.println();

      // new generation becomes last
      System.arraycopy(b, 0, a, 0, b.length);
    }

    System.out.println("end.");
  }
}
```

**In Python:**

```
from __future__ import print_function
import sys, random

n = 16 # number of generations of the colony
a = [] # last generation
b = [] # new generation
useRandom = True

if not useRandom:
  # input a colony string
  print("Enter an initial colony string of 0's and 1's:")
  line = sys.stdin.readline().strip()
else:
  # generate random string
  length = random.randint(2, 50) # random length from 2 to 50 characters
  line = ''
  for i in range(length):
    if random.random() < 0.5: # '0' and '1' are equally weighted (50/50)
      line += '0'
    else:
      line += '1'
  print("random input string: ")
  print(line)

# verify input
if len(line) < 2:
  raise Exception("input length must be >= 2")
for i in range(len(line)):
  if line[i]!='0' and line[i]!='1':
    raise Exception("input must contain only '0' or '1'")

# convert string to integer array
a = [0 for i in range(len(line))]
```

```
b = [0 for i in range(len(line))]
for i in range(len(line)):
  if line[i] == '1':
    a[i] = 1

# produce next generations
for i in range(n):
  b[0] = 0 ^ a[1] # left boundary
  for j in range(1, len(a)-1):
    b[j] = a[j-1] ^ a[j+1]
  b[len(a)-1] = a[len(a)-2] ^ 0 # right boundary

  # print
  for j in range(len(b)):
    if b[j] == 0:
      print(' ', end='')
    else:
      print('x', end='')
  print()

  # new generation becomes last
  a = b[:]

print("end.")
```

## 10 points:

You are given a rectangular area that consists of x's and spaces. You need to write a program that calculates the number of "strands" in this area. A strand is a sequence of x's that are either horizontally or vertically (but **not** diagonally!), connected. For example, here is a single strand:

```
xxxxxxxx
x      x
```

but here there are 3 strands:

```
xx
  xx
    xx
```

and here there are 3 stands as well:

```
xxxx xxxx


        x
```

Note that a single x not connected to anything else is counted as a strand.
Your program should take on input two numbers N and M, N defining the number of lines in the area, and M defining the length of each line, followed by N strings that should consist of x's and spaces. On the output print the number of strands detected.

You can play with the following sample inputs:

1)

```
4 9
xxxx xxxx
   xxx
x   x   x
xxxxxxxxx
```

2)

```
8 11
xx x xx x
x  x xx x
xx   xx  x
xxxxxxxxx x
        xx
xxxxxxxxxxx
x x x x x
  x x x x
```

## Solution:

Here is one of solutions. Let's represent our area as a matrix (a 2-D array). We'll scan it from left to right the top row, then left to right the 2nd row, and so on down to the bottom row. We'll mark the visiting "x" with a number. Every time we encounter a space or the next line, we'll increment the number. We'll also remember the spot where the next strand starts. If we see that the top neighbour has a smaller number then we jump back to where the strand started ("backtracking") and re-mark it with the number of that top neighbour. We'll also decrement other strands' numbers by 1. Here is an example.

Given:

```
xxxx xxxx
   xxx
x   x   x
xxxxxxxxx
```

We start marking the first strand with 1:

```
1111 xxxx
   xxx
x   x   x
xxxxxxxxx
```

We encountered a space, so our current number becomes 2. Continue until the end of line. Increment the current number (to 3).

```
1111 2222
   xxx
x   x   x
xxxxxxxxx
```

On the 2nd line we want to mark the first "x" with 3, however, its top neighbour is 1, so use that number.

```
1111 2222
   111
x   x   x
xxxxxxxxx
```

Now we see that its top neighbour has a bigger number (2) than the current one (1), so we jump back where the strand "2" started and remark.

```
1111 1111
   111
x   x   x
xxxxxxxxx
```

Now on the line 3 we start with "2".

```
1111 1111
   111
2   x   x
xxxxxxxxx
```

We'll use "1" for the next "x" because its top neighbour is "1". The last "x" on the line is marked with "3" since it's the next available number.

```
1111 1111
   111
2   1   3
xxxxxxxxx
```

We start marking the last line with "2"s (because of the top neighbour) until we encounter the middle "x" having "1" as its top neighbour.

```
1111 1111
   111
2   1   3
22221xxxx
```

So we backtrack to "2"s to redo them.

```
1111 1111
   111
1   1   2
111111111
```

And redo the "new" "2"s.

```
1111 1111
   111
1   1   1
111111111
```

Therefore in this example we get only 1 strand.

For another solution we can use a different idea. Again we start with leftmost topmost cell. We'll follow each strand as far as possible by examining the right spot, bottom, top and left ones recursively. The recursion terminates if we hit a space or the boundary or already market spot. Then we move to the right or the next row incrementing our counting number. If we encounter a spot that is already marked with a number we'll skip it. Let's see the same example.

```
xxxx xxxx
   xxx
x   x   x
xxxxxxxxx
```

We go right first then bottom then top then left.

```
1111 xxxx
   xxx
x   x   x
xxxxxxxxx

1111 xxxx
   1xx
x   x   x
xxxxxxxxx

1111 xxxx
   111
x   x   x
xxxxxxxxx

1111 xxxx
   111
x   1   x
xxxx11111

1111 xxxx
   111
x   1   1
xxxx11111

1111 xxxx
   111
x   1   1
111111111

1111 1111
   111
1   1   1
111111111
```

Both algorithms will examine each spot sequentially. And both algorithms will revisit some of the nodes. The difference is that the 2nd algo will skip these cells whereas the 1st one will remark them. In our case the process of remarking is trivial but, for example, for printer it's impossible. The downside of the second algo is that it may run out of stack memory for huge grids and truly complex strands. Let's implement the 2nd algorithm because of the nice property of marking

correctly the 1st time and also because the recursion is a very important technique, so it's worth repeating the exercise (we did it for the pirates treasure).

Please see strand.py and StrandMain.java.

### In Python:

```
from __future__ import print_function
import sys
import re

class Strand:
  a = [[]] # area; use -1 for 'x' and 0 for ' '; we'll use an integer > 0 as a strand ID
  m = 0 # number of columns
  n = 0 # number of rows
  nextId = 1 # next strand number/ID

  def getLastId(self):
    return self.nextId-1

  def readInput(self):
    print("Enter number of rows and columns (separated by space):")
    line = sys.stdin.readline()
    matched = re.match("^\\s*(\\d+)\\s+(\\d+)\\s*$", line)
    if matched:
      self.n = int(matched.group(1))
      self.m = int(matched.group(2))
    else:
      raise Exception("invalid input")

    self.a = [['.' for j in range(self.m)] for i in range(self.n)]

    print("Enter your map with 'x' and ' ' line by line:")
    for i in range(self.n):
      line = sys.stdin.readline().rstrip('\n')
      if len(line) > self.m:
        raise Exception("line must not exceed m")
      chs = list(line)
      j = 0
      for j in range(len(chs)):
        if chs[j] == ' ':
          self.a[i][j] = 0
        elif chs[j] == 'x':
          self.a[i][j] = -1
        else:
          raise Exception("line must contain only 'x' or ' '")

      for j in range(len(chs), self.m):
        self.a[i][j] = 0 # user didn't type white spaces at the end

  def readInputFixed(self):
    """
    self.n = 4
    self.m = 9
    self.a = [[-1,-1,-1,-1, 0,-1,-1,-1,-1],
              [ 0, 0, 0,-1,-1,-1, 0, 0, 0],
              [-1, 0, 0, 0,-1, 0, 0, 0,-1],
              [-1,-1,-1,-1,-1,-1,-1,-1,-1],
             ]
    """
    self.n = 8
    self.m = 11
    self.a = [[-1,-1, 0,-1, 0,-1,-1, 0,-1, 0, 0],
```

```
              [-1, 0, 0,-1, 0,-1,-1, 0,-1, 0, 0],
              [-1,-1, 0, 0, 0,-1,-1, 0, 0,-1, 0],
              [-1,-1,-1,-1,-1,-1,-1,-1,-1, 0,-1],
              [ 0, 0, 0, 0, 0, 0, 0,-1,-1, 0, 0],
              [-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1],
              [-1, 0,-1, 0,-1, 0,-1, 0,-1, 0, 0],
              [ 0, 0,-1, 0,-1, 0,-1, 0,-1, 0, 0]
             ]

  def printArea(self):
    # find the max, so we know how many digits are needed for each cell
    # this is just to keep things aligned
    maxx = max([max(x) for x in strand.a])
    minn = min([min(x) for x in strand.a])

    digits = 1
    maxx /= 10
    while maxx >= 1:
      maxx /= 10
      digits += 1
    if minn < 0:
      digits +=1 # add a space for sign

    fmt = "%{}d".format(digits)
    for i in range(self.n):
      for j in range(self.m):
        print(fmt % self.a[i][j], end='')
      print()
    print()

  def findStrands(self):
    for i in range(self.n):                          # iterate over all cells
      for j in range(self.m):
        if self.a[i][j] == 0:
          continue                                   # empty space; skip
        if self.a[i][j] > 0:
          continue                                   # this cell is already a part of a strand;
skip
        self.a[i][j] = self.nextId                   # start a new strand
        self.followStrand(i, j+1, i, j, self.nextId) # try to follow the strand right
        self.followStrand(i-1, j, i, j, self.nextId) # try to follow the strand up
        self.followStrand(i+1, j, i, j, self.nextId) # try to follow the strand down
        self.nextId +=1                              # increment the strand number
    pass

  def followStrand(self, i, j, prevI, prevJ, currentId):
    if i<0 or i>=self.n or j<0 or j>=self.m:     # out of bounds
      return
    if self.a[i][j] == 0:                        # empty space
      return
    if self.a[i][j] > 0:                         # encountered another strand
      return

    self.a[i][j] = currentId                     # extend the strand

    if i!=prevI or j+1!=prevJ:                   # we don't want to go back where we came from
      self.followStrand(i, j+1, i, j, currentId) # try to follow the strand right

    if i-1!=prevI or j!=prevJ:
      self.followStrand(i-1, j, i, j, currentId) # try to follow the strand up

    if i+1!=prevI or j!=prevJ:
      self.followStrand(i+1, j, i, j, currentId) # try to follow the strand down

    if i!=prevI or j-1!=prevJ:
      self.followStrand(i, j-1, i, j, currentId) # try to follow the strand left
```

```
    pass

############################################################

strand = Strand()
#strand.readInput()
strand.readInputFixed()
print("input area:")
strand.printArea()

strand.findStrands()
print("all strands:")
strand.printArea()
print("number of strands = {}".format(strand.getLastId()))
print("end.")
sys.exit(0)
```

## In Java:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class StrandMain {
  private int[][] a; // area; use -1 for 'x' and 0 for ' '; we'll use an integer > 0 as a
strand ID
  private int m, n;  // m columns, n rows
  private int nextId = 1; // next strand number/ID

  /**
   * @return last used ID which corresponds to the total number of different strands
   */
  public int getLastId() {
    return nextId-1;
  }

  public void readInput() throws Exception {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Enter number of rows and columns (separated by space):");
    String str = br.readLine();
    String[] parts = str.split("\\s");
    if(parts.length != 2)
      throw new Exception("there must be only 2 numbers");
    n = Integer.parseInt(parts[0]);
    m = Integer.parseInt(parts[1]);
    if(n<1 || m<1)
      throw new Exception("number or rows and columns must be > 0");
    a = new int[n][m];

    System.out.println("Enter your map with 'x' and ' ' line by line:");
    for(int i=0; i<n; i++) {
      str = br.readLine();
      if(str.length() > m)
        throw new Exception("line must not exceed m");
      char[] chs = str.toCharArray();
      int j;
      for(j=0; j<chs.length; j++) {
        if(chs[j] == ' ')
          a[i][j] = 0;
        else if(chs[j] == 'x')
          a[i][j] = -1;
        else
          throw new Exception("line must contain only 'x' or ' '");
      }
      for(j=chs.length; j<m; j++)
        a[i][j] = 0; // user didn't type white spaces at the end
```

```
  }
}

/**
 * hard coded input
 */
public void readInputFixed() {
  /*
  n = 4;
  m = 9;
  a = new int[][] {{-1,-1,-1,-1, 0,-1,-1,-1,-1},
                   { 0, 0, 0,-1,-1,-1, 0, 0, 0},
                   {-1, 0, 0, 0,-1, 0, 0, 0,-1},
                   {-1,-1,-1,-1,-1,-1,-1,-1,-1},
                  };
  */
  n = 8;
  m = 11;
  a = new int[][] {{-1,-1, 0,-1, 0,-1,-1, 0,-1, 0, 0},
                   {-1, 0, 0,-1, 0,-1,-1, 0,-1, 0, 0},
                   {-1,-1, 0, 0, 0,-1,-1, 0, 0,-1, 0},
                   {-1,-1,-1,-1,-1,-1,-1,-1,-1, 0,-1},
                   { 0, 0, 0, 0, 0, 0, 0,-1,-1, 0, 0},
                   {-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1},
                   {-1, 0,-1, 0,-1, 0,-1, 0,-1, 0, 0},
                   { 0, 0,-1, 0,-1, 0,-1, 0,-1, 0, 0}
                  };
}

public void printArea() {
  // find the max, so we know how many digits are needed for each cell
  // this is just to keep things aligned
  int max = 0;
  int min = Integer.MAX_VALUE;
  for(int i=0; i<n; i++) {
    for(int j=0; j<m; j++) {
      if(a[i][j] > max)
        max = a[i][j];
      if(a[i][j] < min)
        min = a[i][j];
    }
  }

  int digits;
  for(digits=1, max/=10; max>=1; digits++)
    max /= 10;
  if(min < 0)
    digits++; // add a space for sign

  String format = "%"+Integer.toString(digits)+"d";
  for(int i=0; i<n; i++) {
    for(int j=0; j<m; j++) {
      System.out.printf(format, a[i][j]);
    }
    System.out.println();
  }
  System.out.println();
}

public void findStrands() {
  for(int i=0; i<n; i++) {                // iterate over all cells
    for(int j=0; j<m; j++) {
      if(a[i][j] == 0)
        continue;                         // empty space; skip
      if(a[i][j] > 0)
        continue;                         // this cell is already a part of a strand; skip
```

```
        a[i][j] = nextId;                   // start a new strand
        followStrand(i, j+1, i, j, nextId); // try to follow the strand right
        followStrand(i-1, j, i, j, nextId); // try to follow the strand up
        followStrand(i+1, j, i, j, nextId); // try to follow the strand down
        nextId++;                           // increment the strand number
      }
    }
  }

  private void followStrand(int i, int j, int prevI, int prevJ, int id) {
    if(i<0 || i>=n || j<0 || j>=m)    // out of bounds
      return;
    if(a[i][j] == 0)                  // empty space
      return;
    if(a[i][j] > 0)                   // encountered another strand
      return;

    a[i][j] = id;                     // extend the strand

    if(i!=prevI || j+1!=prevJ)        // we don't want to go back where we came from
      followStrand(i, j+1, i, j, id); // try to follow the strand right

    if(i-1!=prevI || j!=prevJ)
      followStrand(i-1, j, i, j, id); // try to follow the strand up

    if(i+1!=prevI || j!=prevJ)
      followStrand(i+1, j, i, j, id); // try to follow the strand down

    if(i!=prevI || j-1!=prevJ)
      followStrand(i, j-1, i, j, id); // try to follow the strand left
  }

  public static void main(String[] args) throws Exception {
    StrandMain strand = new StrandMain();
    //strand.readInput();
    strand.readInputFixed();
    System.out.println("input area:");
    strand.printArea();

    strand.findStrands();
    System.out.println("all strands:");
    strand.printArea();
    System.out.println("number of strands = "+strand.getLastId());
    System.out.println("end.");
  }
}
```