PROBLEM OF THE MONTH

January, 2020

2019 - 2020

MATHEMATICS

5 points:

One day the Little Red Riding Hood decided to take some pies to her grandmother's house. Shortly after leaving her house, however, she realized she was really hungry and ate the three largest pies, and as a result the total weight of the pies decreased by 35%. A little bit later in her journey, she once again wanted a snack and ate the three smallest pies, as a result of which the pies' total weight decreased by another 5/13 compared to the previous weight. How many pies did the Little Red Riding Hood leave her house with?

Answer: 10

Solution:

The three largest pies weigh 35% of the combined weight, or 7/20, leaving 13/20. Later on, the Little Red Riding Hood ate 5/13 of the remaining 13/20 of the pies. 5/13 out of 13/20 is 5/20, or 25%.
So we have that the three largest pies weighed 35% of the total (11.67% average weight) and the three smallest weighed 25% of the total (8.33% average weight). The remaining pies must then add up to 40% of the total weight and their average weight must be between 8.33% and 11.56% of the total. The only possibility then is that there are 4 pies remaining, so the number of pies the Little Red Riding Hood left her grandmother's house with is 3+3+4 = 10.

10 points:

The line l intersects the segment AB at point D so that $AD = a$ and $BD = b$. Construct a circle that goes through points A and B and carves out the smallest possible segment from line l. What is the length of this segment in terms of a and b?

Hint: use the chord theorem

Answer: $2\sqrt{ab}$

Solution: The point Ddivides the carved out segment into two segments. Let us denote the lengths of these segments c and d. By the chord theorem $cd = ab$. Our goal is to minimize $c + d$ given their product or, equivalently, minimize the arithmetic mean of c and d if their geometric mean is given. It is well known (and easy to prove) that this happens when $c = d = \sqrt{ab}$. Therefore, the minimal length of the segment is $2\sqrt{ab}$. Remarkably, it does not depend on the angle at which l intersects AB. To construct the circle we notice that the point Ddivides the carved out segment into two equal segments. We draw the perpendicular to l from the point D and the perpendicular bisector to AB. The center of the circle is at the intersection of these constructed lines.

PHYSICS

5 points: Imagine that you don't have a freezer. There is an easy trick that would allow you to achieve a rather low temperature: mix salt and ice. As salt causes the ice to melt, the temperature of the mixture drops. This way, the melting temperature of ice can be brought down to as low as $T_1 = -21^0C$. What fraction of ice will be melted when that temperature is achieved if you started with ice at its regular melting point, $T_0 = 0^0C$? Assume the system to be thermally isolated. Specific heat capacities of liquid water and ice are $C_w = 4.2\frac{kJ}{kg\cdot{}^0C}$■ and $C_i = 2.1\frac{kJ}{kg\cdot{}^0C}$■, respectively. Neglect heat capacity of salt. Latent heat of ice melting is L=334 kJ/kg.

Hint: Note that the latent heat of molten ice would be provided by cooling both remaining ice and salty water to a lower temperature.

Answer: approximately 15%.

Solution: Let □ be the original mass of ice, and $x \cdot M$ the mass that has melted. The amount of heat needed for the melting is xML, and use of this heat result in cooling of both ice and water from $T_0 = 0^0C$ to $T_1 = -21^0C$:

$$xML = (C_w xM + C_i(1-x)M)(T_0 - T_1)$$
$$334x = 21 \cdot (4.2x + 2.1(1-x))$$

By solving this equation, we obtain x=0.15, or approximately 15%.

10 points: Commercial heat pads are based on the solution of sodium acetate. Combined with water, this salt may form a crystal with a melting temperature close to $T_m = 58^oC$. This crystal will not typically form spontaneously when the solution is cooled down starting from higher temperatures. In other words, the liquid would remain "supercooled" below the melting point. The crystallization can be triggered by local mechanical stress, leading to heating up of the solution. Assume that the heat pad of certain mass M is placed into a thermally isolated container that contains the same mass of water, at temperature $T_0 = 20C$. What will be the final temperature inside that container, once the crystallization is triggered? Specific heat capacities of water and the sodium acetate solution are $C_w = 4.2\frac{kJ}{kg\cdot{}^0C}$■ and $C_s = 3\frac{kJ}{kg\cdot{}^0C}$■, respectively. Ignore the mass and heat capacity of all the parts of the head pad except for the sodium acetate solution itself. The composition of the solution is such that it can crystallize completely at T_m, the latent heat of its crystallization is L=270 kJ/kg.

Hint: First, assume that all of the sodium acetate gets crystallized, and find how much heat this gives out. This determines the change in Temperature. Once you've done that, change if your assumption is correct (is the temperature that you found below or above the melting point?)

Answer: $T = 57.5\,^{o}C$

Solution: Let M be the mass of the sodium acetate in the heat pack (the mass of water is the same). If we assume that all of it is crystallized, that would produce extra heat LM. This heat would change the temperature of both water and sodium acetate by amount

$$\Delta T = \frac{LM}{M(C_s + C_w)} = L/(C_s + C_w) = 37.5\,^{o}C$$

The resulting temperature is $(37.5{+}2)\,^{o}C = 57.5\,^{o}C$. Since this is slightly below the equilibrium melting point, the sodium acetate has indeed completely crystalized (as assumed).

CHEMISTRY

5 points:

One of Sigma workshops is “Fruit electricity”. By using two different metals, electricity is produced from lemons, apples or other juicy fruits. For the first time, a copper wire and iron nails (galvanized common nails obtained from the Home Depot) were used, and the voltage obtained from the single fruit cell was approximately 1V. For the second time, a copper and X-acto knife blades were used, and it produced about 0.7V. How can you explain this significant difference?

Hint:

A key word here is “galvanized”. What exactly does it mean: “galvanized nails”? Read about that, and that will give an answer.

Solution:

In a bulk metal, atoms lose their outer electrons, which are freely travelling across the whole piece of metal. The atoms become positively charged, and they stay in the nodes of metal’s crystal lattice which is stabilized by the “electron gas” formed by travelling electrons. When a piece of metal contacts water or an aqueous solution, a very tiny fraction of metal atoms dissolves, however, only positive metal ions dissolve, whereas the electrons stay in the metal. As a result, a piece becomes negatively charged, so dissolved ions cannot travel too far, they are attracted to the metal piece by electrostatic forces. As a result, a piece of metal becomes covered by a positive “ion coat”, whereas a metal itself becomes negatively charged.

Electropositive metals, e.g. zinc, lose electrons easily, their ions are easy to form, and a relatively big number of ions go into a solution. Zinc’s “ion coat” is thick, and the voltage between the ionic layer and the piece of zinc metal is high. Copper donates electrons more reluctantly, its “ion coat” is thin, and its positive charge is small, and the negative charge of a piece of copper is small. Iron is less electropositive than zinc, but more electropositive than copper. Now imagine both copper and zinc are placed in the same solution of some weak acid (an acid is needed just to make water more conductive). What happens? Ion coats of both pieces are now connected by a electric conductor (an acidic water), so their electric potentials become equal. *However, we know that voltages between zinc ion coat and zinc metal, and between copper ion coat and copper metal are different*. Taking into account that both ion coats are electrically connected, we will observe that the pieces of zinc and copper have different electric potentials, and if we connect them together, electric current will flow from the electron rich piece of metal (zinc) to the electron deficient piece of metal (copper). Since zinc loses electric charge, the ions from its ion coat escape, and new ions dissolve to compensate for the loss of ions in the ion coat. That produces more electrons that flow to the piece of copper, and this process will last until the whole piece of zinc dissolves.

Clearly, the nail coated with zinc behaves like zinc, and, when connected to copper, they produce higher voltage than less electronegative iron.

In addition, x-acto knife blades are not a pure iron, they contain carbon. However, this carbon forms tiny crystals of “cementite” (a compound formed by four atoms of iron and one atom of carbon). Cementite is very hard, but it is fragile, so a real steel contains less than 4:1 ration (by mole) of iron and carbon. Small grains of cementite are surrounded by iron. That makes steel

elastic (due to iron) and hard (due to cementite). However, for us that is not important, because from the chemistry's point of view, steel and iron are essentially the same.
To summarize, a copper-and-galvanized-nail couple produces higher voltage because it is actually a Cu-Zn couple, whereas a copper-X-acto-blade is a Cu-Fe couple.

10 points:

A coin made from a copper-silver alloy with unknown silver content weighs 1 gram. To this coin, an excess of nitric acid was added, and when the reaction had ceased (all solids "dissolved"), the clear and transparent liquid obtained was evaporated to dryness. The solid remainder was dissolved in 100 ml of water, and 10 grams (excess) of sodium sulfide was added. The precipitate formed was filtered and completely dried. The mass of the residue was 1.38 grams. What is the percentage of silver in the alloy?

Hint:

Both copper and silver form insoluble sulfides, CuO and Ag_2O, respectively. Percentage of sulfur, by mass, is different in them.

Solution:

Both copper and silver react with nitric acid, and the products are copper and silver nitrates, accordingly. They are soluble and non-volatile, so they can be obtained in a free form when a nitric acid evaporates.
When copper nitrate or silver nitrate react with sodium sulfide, they both form insoluble sulfides.

$$Cu(NO_3)_2 + Na_2S \rightarrow CuS + 2\ NaNO_3$$

$$2\ AgNO_3 + Na_2S \rightarrow Ag_2S + 2\ NaNO_3$$

Now we have to calculate the amount of sulfides (by mass) produced from 1 gram of each metal. Atomic weights of Cu, S, and Ag are 64, 32, and 108, accordingly, and silver sulfide contains just ½ of sulfur atom per one silver atom. That means 0.64 g of copper yields 0.96 g of sulfide, and 1 g of copper yields 0.96/0.64= 1.5 g of sulfide. Accordingly, 1.08 g of silver yields 1.24 g of sulfide, and 1 g of silver yields 1.15 g of sulfide.
Our coin yielded 1.38 g of a mixture of copper and silver sulfides, and this information allows us to calculate the silver content (by mass). If we denote it as x, the rest, or $1-x$, is copper. That allows us to draw an equation:

$$1.15x + 1.5(1-x)=1.38$$

It is easy to solve it (this is a Chemistry problem, not Math, so I skip that step), and we get, approximately, 0.34, which means the coin contains around 66% of copper and 34% of silver.

BIOLOGY

5 points:

Why do birds and mammals maintain a constant temperature? A small bird consumes substantially more food than a cold-blooded lizard of the same weight, which warms in the sun during the day and cools again at night.

Answer:

1) Birds and mammals can be active at night and during cold weather -- they can explore environmental niches from which reptiles are barred.
2) Numerous temperature-sensitive metabolic reactions can be better coordinated if they operate in a relatively uniform temperature range.
3) High temperature, besides increasing the rate of chemical reactions, allows faster diffusion. Heat speeds up the diffusion of chemical transmitters in nerves and facilitates faster behavioural reactions. It has also been suggested that maintaining a constant high temperature in the brain promotes memory and learning.

10 points:

Viruses do not only parasitize cells causing their death, but also facilitate horizontal gene transfer between organisms and may even define biogeochemical cycles. Imagine that all the viruses on Earth disappeared at once. What environmental (short-term) and evolutionary (long-term) consequences will this lead to?

Answer:

The virus is a small infectious agent that replicates only inside the living cells of an organism. Viruses can infect all types of life forms, from animals and plants to microorganisms, including bacteria and archaea. The viruses are the most abundant and diverse species on Earth. Every species has its own viruses controlling its population. Therefore, viruses are important for determining the size of the majority of populations of living species, and are the major factor in the ecological balance. The bacterial and microbial biomass of the oceans is controlled by the viruses that infect these marine species. The marine bacterial biomass, in turn, is critical for the amount of food available for the marine life, as well as for the exchange of oxygen, nitrogen, carbon, etc. in the atmosphere. By lysing (killing) the bacteria, viruses release the carbon and nitrogen into the ocean and the atmosphere, thereby controlling the atmospheric composition and temperature.

The viruses are, by design, able to take up genes from one organism and transfer them to the other organism. As such, the viruses appear to be the major driving force in evolution, enriching the genomes of higher organisms via the so called "horizontal gene transfer". As was discovered recently, the evolution happens primarily not via the point mutations of individual nucleotides in DNA, but rather by the organisms acquiring the whole new genes and the corresponding new functions due to the gene transfer from other species by the viruses. The examples include evolution of the novel lineages of key photosynthetic genes in marine microorganisms. Another example is the human (and other mammals) gene that is responsible for the attachment of the fetus to the mother's placenta. This protein called Hemo is produced by the fetus, and it comes from the viral gene acquired by our ancestors about 100 millions years ago. There are many other genes of viral origin that constitute up to 8% of our genome with yet unknown functions.

Therefore, if all viruses were to disappear at once, we would find the world very different from what it is now both immediately (short term), as well as in its ability to evolve (long term). The suppression of some species by the viruses over the other ones will immediately affect the balance between the population of different species, leading to the extinction of some of them, and the dominance of the other ones. The balance in the ecosystem will be disturbed, most likely leading to ecological collapse. Also, the exchange of nutrients and major chemical elements through the ocean would change, leading to the major change in the atmospheric composition and the climate shift. In the long term, we would find that the disappearance of the viruses leads to the major slowing down of all evolutionary processes.

COMPUTER SCIENCE

- You can write and compile your code here: http://www.tutorialspoint.com/codingground.htm
- Your program should be written in Java or Python
- No GUI should be used in your program: eg., easygui in Python. All problems in POM require only text input and output. GUI usage complicates solution validation, for which we are also using *codingground* site. Solutions with GUI will have points deducted or won't receive any points at all.
- Please make sure that the code compiles and runs on http://www.tutorialspoint.com/codingground.htm before submitting it.
- Any input data specified in the problem should be supplied as user input, not hard-coded into the text of the program.
- Submit the problem in a plain text file, such as .txt, .dat, etc. **No .pdf, .doc, .docx, etc!**

The Game of Hungry Sigma-Fish

You are given a rectangular field throughout which a number of hungry Sigma-fish is spread out. Each fish is denoted by an integer representing its size. Empty fields are represented by a period "." Here is an example of a simple field:

```
..4....3
.....2..
.2......
.......1
```

Each move goes like that: each Sigma-fish [simultaneously] moves one field (vertically, horizontally or diagonally) towards another Sigma-fish of smaller size (if any). The direction is chosen toward the closest smaller fish, with the following tie-breakers:

- larger target has a priority
- in case of equal size targets or alternative fields to move, the priority is directional, in the following order of decreasing priority: N, NE, E, SE, S, SW, W, NW.

At the end of each move, if two or more fish end up on the same field, they merge into a fish with the combined size (sum of their previous sizes).

5 points:

For this problem the game field is a square NxN.

Write a program that will receive on the input:

- size N of the field (optionally)
- field composition, which consists of periods and integers

Your program should determine whether each of the integers 1..N is found in the set of coordinates of the fish on the board and print YES or NO correspondingly. In case of NO, as a bonus, please print which integers are missing.

Note: coordinates start with 1, thus the coordinates of the top left field of the board is (1, 1), and the coordinates of the bottom right field of the board is (N, N).

Solution:

Python:

```
"""
Assumptions:
*) if the 1st line is a number, it's N (not a field 1x1 with a fish number N)
*) if N is given, the field has to measure to it
   else N is determined from the width of the 1st row
*) we are asked to find if there are any empty rows or columns in the field table
"""

import re

def parse_line(line):
  a = []
  res = re.search(r"^(\.|[+-]?\d+)+$", line) # check if the line consists only of dot(s) or
integer(s)
  if res:
    res = re.findall(r"\.|[+-]?\d+", line) # find all such constituents
    for r in res:
      # print(r)
      a.append(r)
  else:
    raise Exception("invalid input")
  return a

field = []
line = input("Enter N or the first row: ").strip()
res = re.match("^(\d+)$", line)
if res:
  n = int(res.group(0))
  m = n # number of rows to read
else:
  a = parse_line(line)
  n = len(a)
  m = n - 1
  field.append(a)
```

```
# read other rows
for i in range(m):
  line = input("Enter row %d: " % (i+1+n-m)).strip()
  a = parse_line(line)
  if len(a) == n:
    field.append(a)
  else:
    raise Exception("invalid input")

# reset number of rows
m = len(field)

# collect coordinates
c = set()
for i in range(m):
  for j in range(n):
    if field[i][j] != '.':
      c.add(i)
      c.add(j)

# see if anything is missing
for i in range(n):
  if i in c:
    print("%d - YES" % (i+1))
  else:
    print("%d - NO" % (i+1))

print("end.")
exit(0)
```

Java:

```
/*
Assumptions:
*) if the 1st line is a number, it's N (not a field 1x1 with a fish number N)
*) if N is given, the field has to measure to it
   else N is determined from the width of the 1st row
*) we are asked to find if there are any empty rows or columns in the field table
*/

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Set;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Fish5 {
  private String[][] field;
  private int m; // number of rows in field
  private int n; // number of columns in field

  String[] parseLine(String line) throws Exception {
    ArrayList<String> a = new ArrayList<>();
    // you can optimize the 2 regex by incrementally parsing into '.' and integers
    // but I use regex here for clarity
    if(line.matches("^(\\.|[+-]?\\d+)+$")) { // check if the line consists only of dot(s) or 
integer(s)
      Pattern pattern = Pattern.compile("\\.|[+-]?\\d+"); // find all such constituents
      Matcher matcher = pattern.matcher(line);
      while(matcher.find()) {
        a.add(matcher.group());
      }
    }
    else
      throw new Exception("invalid input");
    String[] dummy = new String[0];
    return a.toArray(dummy);
  }

  void input() throws Exception {
    System.out.print("Enter N or the first row: ");
    BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
    String line = reader.readLine().trim();
    Matcher matcher = Pattern.compile("^(\\d+)$").matcher(line);
    int k = 0; // number of rows to read
    if(matcher.find()) {
      n = Integer.parseInt(matcher.group());
      m = n;
      k = m;
      field = new String[m][n];
    }
    else {
      String[] a = parseLine(line);
      n = a.length;
      m = n;
      k = n - 1;
      field = new String[m][n];
      System.arraycopy(a, 0, field[0], 0, n);
    }

    // read other rows
    for(int i = n - k; i < m; i++) {
      System.out.printf("Enter row %d: ", i + 1);
      line = reader.readLine().trim();
      String[] a = parseLine(line);
      if(a.length == n)
        System.arraycopy(a, 0, field[i], 0, n);
      else
        throw new Exception("invalid input");
    }
  }

  void solve() {
    // collect coordinates
    Set<Integer> c = new HashSet<>();
    for(int i = 0; i < m; i++) {
      for(int j = 0; j < n; j++) {
        if(!field[i][j].equals(".")) {
          c.add(i);
          c.add(j);
        }
      }
    }

    // see if anything is missing
    for(int i=0; i<n; i++) {
      if(c.contains(i))
        System.out.printf("%d - YES\n", i + 1);
      else
        System.out.printf("%d - NO\n", i + 1);
    }
  }
  public static void main(String[] args) throws Exception {
    Fish5 fish = new Fish5();
    fish.input();
    fish.solve();
  }
}
```

10 points:

Write a program that will receive on the input:

- size of the Game of Hungry Sigma-Fish field as integers N and M (optionally)
- field composition, which consists of periods and integers

Your program should print the final field state. You may optionally elect to print all intermediate field states. Finally, one point will be reserved for the quality of presentation.

Solution:

Python:

```
"""
Assumptions:
*) if the 1st line is 2 numbers, they are m (number of rows) and n (number of columns)
*) if the 1st line is 1 number, it's m (number of rows); n is deducted from the 1st row
*) if n is given, the field has to measure to it
*) the smallest fish does not swim
*) game is over when nobody swims
"""

import re
import math

def parse_line(line):
  a = []
  res = re.search(r"^(\.|[+-]?\d+)+$", line) # check if the line consists only of dot(s) or integer(s)
  if res:
    res = re.findall(r"\.|[+-]?\d+", line) # find all such constituents
    for r in res:
      a.append(r)
  else:
    raise Exception("invalid input")
  return a

field = []
line = input("Enter M, N or just M: ").strip()
numbers_str = re.split(r"[\s,]\s*", line)
# the following will croak if the elements are not all integers
numbers = [int(X) for X in numbers_str]
# also verify that they all > 0
assert(all(x > 0 for x in numbers))
if len(numbers) == 1:
  m = numbers[0]
elif len(numbers) == 2:
  m = numbers[0]
  n = numbers[1]
  k = m # lines to read
else:
  raise Exception("1st line must have either 1 or 2 integers")

if len(numbers) == 1:
  line = input("Enter row 1: ").strip()
  a = parse_line(line)
  n = len(a)
  field.append(a)
  k = m - 1 # lines to read

# read other rows
for i in range(k):
  line = input("Enter row %d: " % (i+1+m-k)).strip()
  a = parse_line(line)
  if len(a) == n:
    field.append(a)
  else:
    raise Exception("invalid input")

# a more efficient way would be to update the width only when 1 fish eats another
# but itâ€™s separated here for clarity
def get_width(field):
  width = 1
  m = len(field)
  if m > 0:
    n = len(field[0])
    for i in range(m):
      for j in range(n):
        w = len("%s" % field[i][j])
        if w > width:
          width = w
  return width + 1 # +1 to always separate columns

def print_aquarium(field):
  m = len(field)
  if m > 0:
    n = len(field[0])
    format = "%%%ds" % get_width(field)
    for i in range(m):
      for j in range(n):
        print(format % field[i][j], end="")
      print()
  print()

# according to stated preference (from North clockâ€”wise)
def circumference(r):
  indices = []
  for i in range(r):
    indices.append((-r,i))
  for i in range(-r,r):
    indices.append((i,r))
  for i in range(r,-r,-1):
    indices.append((r,i))
  for i in range(r,-r,-1):
    indices.append((i,-r))
  for i in range(-r,0):
    indices.append((-r,i))
  # print("circumference(%d) = " % r, indices)
  return indices
```

```
def find_closest_fish(i, j, v, field, m, n):
  for r in range(1,max(m,n)): # make search radius progressively larger
    found = ()
    for (dx, dy) in circumference(r):
      # print("dx=%d, dy=%d" % (dx, dy))
      if i+dx<0 or i+dx>=m or j+dy<0 or j+dy>=n or field[i+dx][j+dy]=='.':
        continue
      # print(â€œfound fish %d at (%d, %d)" % (int(field[i+dx][j+dy]), i+dx, j+dy))
      if int(field[i+dx][j+dy]) < v: # smaller fish than I
        if len(found) == 0:
          found = (i+dx,j+dy)
        else:
          if int(field[i+dx][j+dy]) > int(field[found[0]][found[1]]): # bigger fish than previously
found
            found = (i+dx,j+dy) # keep only largest
    if len(found) > 0:
      return found
    else:
      # print("debug: extending radius")
      pass
  return ()

def move_fish(field):
  m = len(field)
  if m > 0:
    n = len(field[0])
    new_field = [['.' for i in range(n)] for j in range(m)]
    for i in range(m):
      for j in range(n):
        if field[i][j] != '.':
          z = find_closest_fish(i, j, int(field[i][j]), field, m, n)
          if len(z) == 2:
            (x, y) = z
            # print("found fish %d at (%d, %d)" % (int(field[x][y]), x, y))
            den = math.sqrt((x-i)**2 + (y-j)**2)
            new_i = i + round((x-i) / den)
            new_j = j + round((y-j) / den)
          else: # smallest fish does not swim
            new_i = i
            new_j = j
          if new_field[new_i][new_j] == '.':
            new_field[new_i][new_j] = field[i][j]
          else:
            new_field[new_i][new_j] = int(new_field[new_i][new_j]) + int(field[i][j])
    return new_field
  else:
    return []

# field = [['.', '.', '.', '.', '.', '.', '.', '.'],
#          ['.', '2', '3', '.', '.', '.', '.', '.'],
#          ['.', '1', '4', '.', '.', '.', '.', '.'],
#          ['.', '.', '.', '.', '.', '.', '.', '.']
#         ]
# m = len(field)
# n = len(field[0])

for t in range(n**3):
  print_aquarium(field)
  new_field = move_fish(field)
  if new_field == field:
    # nothing swam; stop
    break
  else:
    field = new_field
print("end.")
exit(0)
```

Java:

```
/*
Assumptions:
*) if the 1st line is 2 numbers, they are m (number of rows) and n (number of columns)
*) if the 1st line is 1 number, it's m (number of rows); n is deducted from the 1st row
*) if n is given, the field has to measure to it
*) the smallest fish does not swim
*) game is over when nobody swims
*/

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

class Coordinate {
  int x, y;

  Coordinate(int x, int y) {
    this.x = x;
    this.y = y;
  }
}

public class Fish10 {
  private String[][] field;
  private int m; // number of rows in field
  private int n; // number of columns in field

  String[] parseLine(String line) throws Exception {
    ArrayList<String> a = new ArrayList<>();
    // you can optimize the 2 regex by incrementally parsing into '.' and integers
    // but I use regex here for clarity
    if(line.matches("^(\\.|[+-]?\\d+)+$"))  {  //  check  if  the  line  consists  only  of  dot(s)  or
integer(s)
      Pattern pattern = Pattern.compile("\\.|[+-]?\\d+"); // find all such constituents
      Matcher matcher = pattern.matcher(line);
      while(matcher.find()) {
        a.add(matcher.group());
      }
    }
    else
      throw new Exception("invalid input");
    String[] dummy = new String[0];
    return a.toArray(dummy);
  }

  void input() throws Exception {
    System.out.print("Enter M, N or just M: ");
    BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
    String line = reader.readLine().trim();
    String[] nums = line.split("\\s*[\\s,]\\s*");
    // the following will croak if the elements are not all integers
    int[]             numbers             =             Arrays.stream(nums).map(s              ->
Integer.valueOf(s)).mapToInt(Integer::intValue).toArray();
    // also verify that they all > 0
    if(!Arrays.stream(numbers).allMatch(i -> i > 0))
      throw new AssertionError("all numbers must be positive");
    int k = 0; // lines to read
    if(numbers.length == 1)
      m = numbers[0];
    else if(numbers.length == 2) {
      m = numbers[0];
      n = numbers[1];
      field = new String[m][n];
      k = m; // lines to read
    }
    else
      throw new Exception("1st line must have either 1 or 2 integers");

    if(numbers.length == 1) {
      System.out.print("Enter row 1: ");
      line = reader.readLine().trim();
      String[] a = parseLine(line);
      n = a.length;
      field = new String[m][n];
      System.arraycopy(a, 0, field[0], 0, n);
      k = m - 1; // lines to read
    }

    // read other rows
    for(int i=m-k; i<m; i++) {
      System.out.printf("Enter row %d: ", i + 1);
      line = reader.readLine().trim();
      String[] a = parseLine(line);
      if(a.length == n)
        System.arraycopy(a, 0, field[i], 0, n);
      else
        throw new Exception("invalid input");
    }
  }

  // a more efficient way would be to update the width only when 1 fish eats another
  // but it's separated here for clarity
  int getWidth() {
    int width = 1; // of a column for pretty printing
    if(m > 0) {
      for(int i=0; i<m; i++) {
        for(int j=0; j<n; j++) {
          int w = String.format("%s", field[i][j]).length();
          if(w > width)
            width = w;
        }
      }
    }
    return(width+1); // +1 to always separate columns
  }

  void printAquarium() {
    if(m > 0) {
      String format = String.format("%%%ds", getWidth());
      for(int i=0; i<m; i++) {
        for(int j=0; j<n; j++)
```

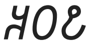
```
          System.out.printf(format, field[i][j]);
        System.out.println();
      }
    }
    System.out.println();
  }

  // according to stated preference (from North clockâ€”wise)
  ArrayList<Coordinate> circumference(int r) {
    ArrayList<Coordinate> indices = new ArrayList<>();
    for(int i=0; i<r; i++)
      indices.add(new Coordinate(-r,i));
    for(int i=-r; i<r; i++)
      indices.add(new Coordinate(i,r));
    for(int i=r; i>-r; i--)
      indices.add(new Coordinate(r,i));
    for(int i=r; i>-r; i--)
      indices.add(new Coordinate(i,-r));
    for(int i=-r; i<0; i++)
      indices.add(new Coordinate(-r,i));
    return indices;
  }

  Coordinate findClosestFish(int i, int j, int v) {
    for(int r=1; r<Math.max(m,n); r++) { // make search radius progressively larger
      Coordinate found = null;
      for(Coordinate c : circumference(r)) {
        int dx = c.x;
        int dy = c.y;
        if(i+dx<0 || i+dx>=m || j+dy<0 || j+dy>=n || field[i+dx][j+dy].equals("."))
          continue;
        if(Integer.parseInt(field[i+dx][j+dy]) < v) { // smaller fish than I
          if(found == null)
            found = new Coordinate(i+dx, j+dy);
          else {
            if(Integer.parseInt(field[i+dx][j+dy]) > Integer.parseInt(field[found.x][found.y])) //
bigger fish than previously found
              found = new Coordinate(i+dx, j+dy); // keep only largest
          }
        }
      }
      if(found != null)
        return found;
    }
    return null;
  }

  String[][] moveFish() {
    if(m > 0) {
      String[][] newField = new String[m][n];
      Arrays.stream(newField).forEach(a -> Arrays.fill(a, "."));
      for(int i=0; i<m; i++) {
        for(int j=0; j<n; j++) {
          if(!field[i][j].equals(".")) {
            Coordinate z = findClosestFish(i, j, Integer.parseInt(field[i][j]));
            int new_i = i;
            int new_j = j;
            if(z != null) {
              int x = z.x;
              int y = z.y;
              double den = Math.sqrt((x - i)*(x - i) + (y - j)*(y - j));
              new_i = i + (int)Math.round((x - i) / den);
              new_j = j + (int)Math.round((y - j) / den);
            }
            else ; // smallest fish does not swim
            if(newField[new_i][new_j].equals("."))
              newField[new_i][new_j] = field[i][j];
            else
              newField[new_i][new_j] = Integer.toString(Integer.parseInt(newField[new_i][new_j]) +
Integer.parseInt(field[i][j]));
          }
        }
      }
      return newField;
    }
    else
      return new String[0][0];
  }

  void solve() {
    for(int t=0; t<n*n*n; t++) {
      printAquarium();
      String[][] newField = moveFish();
      if(Arrays.deepEquals(newField, field)) // nothing swam; stop
        return;
      else
        field = newField;
    }
    System.out.println("it seems that I cannot find a solution");
  }

  public static void main(String[] args) throws Exception {
    Fish10 fish = new Fish10();
    fish.input();
    fish.solve();
    System.out.println("end.");
  }
}
```

LINGUISTICS

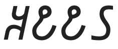
5 points:

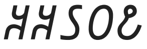
Dr. Sigmund Sigman, a former Sigma camper and current linguist, recently discovered an advanced civilisation of hominids living in ice caves near the Sigmen Glacier in Antarctica. The hominids, who call themselves Sigmamen, use squid ink to write on seal skins. Dr. Sigman had trouble translating the writing system of the natives, especially its numerical calculations, in part because there allegedly were multiple correct representations of the same number. However, after learning to verbally communicate, he could agree with the natives on some common numerical values, which are written below in the Sigmaman number system:

Number of fingers on two hands:
V

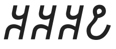
Variant 2: 

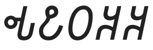
Number of letters in the English alphabet:

Variant 1: 

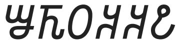
Variant 2: 

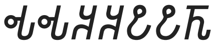
Number of U.S. states:

Variant 1: 

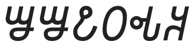
Variant 2: 

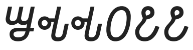
Number of U.N. representatives:

Variant 1: 

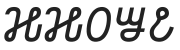
Variant 2: 

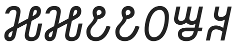
Number of days in a year:

Variant 1: 

Variant 2: 

Current year:

Variant 1: 

Variant 2: 

Given the following examples, help Dr. Sigman by doing the following:

Question 1: Determine the decimal translation of each of the Sigmaman numerals used in the examples. Express your answers in Hindu-Arabic numerals (ex: 0, 1, 2, 3) and/or explain your answer in words.
Question 2: Using the Sigmaman number system to write TWO different variants of:
a. The number of letters in the Russian alphabet
b. The number of representatives in the E.U. parliament

Don't forget to explain how you found your answer!

Answer:

Starting with the number of fingers, which is 10, one can deduce that ℰ=5. Now, if we assume that *0* means subtraction, we can get that Ꮋ=15. Number of English letter = 26, so ᏋℰℰᏚ=26, and that gives us Ꮪ=1. Notice that it is consistent with variant 2, which writes 26 as 15+15+1-5. Similarly, variant 1 for number of US states (50) is consistent with the values above. That also allows us to get another digit's value: ᏺℰ0ᎻᎻ=50, therefore ᏺ+5-15-15=50, and ᏺ=75. Number of UN representatives is 193=ᏺᏺᎻᎻℰℰℏ=75+75+15+15+5+5+ℏ=190+ℏ, and therefore ℏ=3. Further, 193=Ꮚℏ0ᎻᎻℰ=Ꮚ+3-15-15-5=Ꮚ+3-35, and Ꮚ=225. We can now check that both expressions for the number of days in a year are consistent with our findings. The last task is to figure out the value of ℋ. ℋℋ0Ꮚℰ=2020, so 2*ℋ-225-5=2020, 2*ℋ=2250, and ℋ=1125. Now we can verify that ℋℋℰℰ0ᏊᎻ=1125+1125+5+5-225-15=2020.

Question 1:

Ꮪ=1

ℏ=3

ℰ=5

Ꮋ=15

ᏺ=75

Ꮚ=225

ℋ=1125

Question 2:

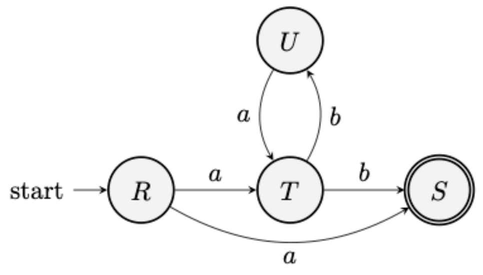
a. Number of letters in Russian alphabet=33=ᎻᎻℏ=ᏺ0ᎻᎻℰℰᏚᏚ (=15+15+3=75-15-15-5-5-1-1)
b. Number of representatives in EU parliament:
   Pre-Brexit: 751=ᏊᏊᏊᏺᏚ=ℋᏚ0ᏺᏺ
   Post-Brexit: 705=ᏊᏊᏊᎻᎻ=ℋᎻᎻ0ᏊᏊ

10 points:

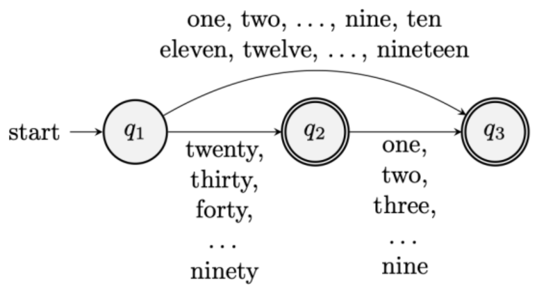
Finite state automata (FSA) are graphs used for recognizing whether some string is valid (*"grammatical"*) in some language. To check whether a string is valid, we start in a vertex of the

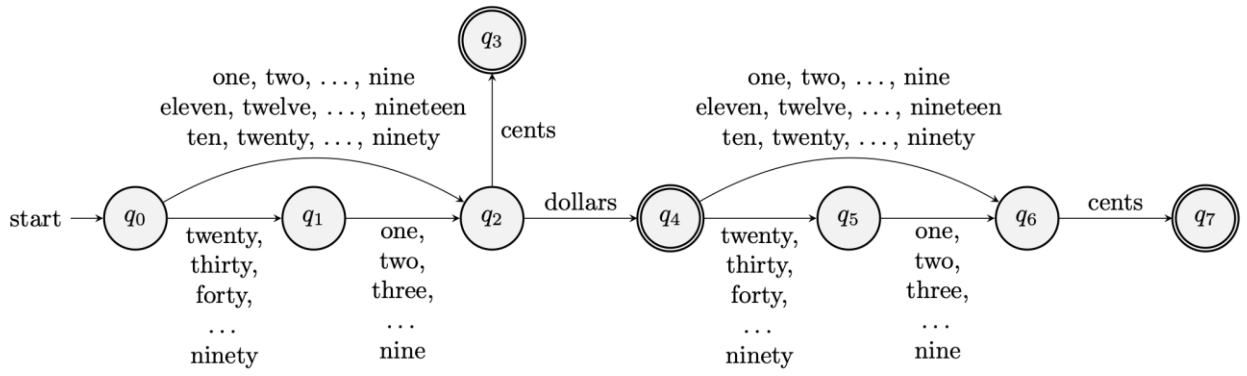
FSA marked with a “start”, and proceed to subsequent vertices based on the elements in the string, reading them one after another. If we finished the string and ended up in a vertex marked with a double circle (such vertices are called *“final states”*), the input string is *grammatical* in a given language. If we exhausted the string but didn’t end up in a final state, the string is *ungrammatical*. Similarly, if we encountered an element in the string which does not have a corresponding arrow going out of the vertex we are currently in, the string is also *ungrammatical*.

For example, for the automaton below, the following strings are grammatical: *a*, *ab*, *abab*, *ababab*, etc., and the following strings are ungrammatical: *b*, *aa*, *ba*, *bb*, *aba*, *abb*, *abaa*, etc.



FSAs are often used in natural language processing. The two examples below show FSAs recognizing English phrases for numbers from 1 to 99 and for prices in dollars and cents under $100, respectively.

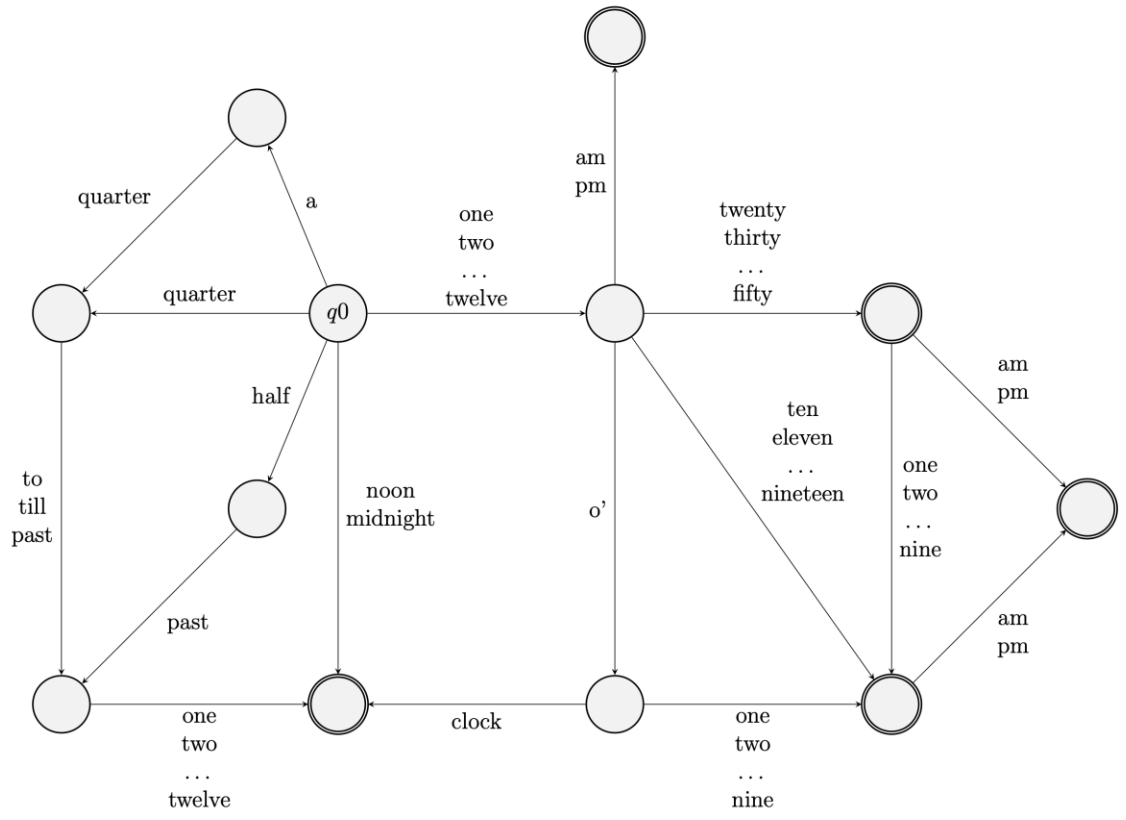

one, two, . . . , nine, ten
eleven, twelve, . . . , nineteen
start
q1
q2
q3
twenty, thirty, forty, . . . ninety
one, two, three, . . . nine


q3
one, two, . . . , nine
eleven, twelve, . . . , nineteen
ten, twenty, . . . , ninety
cents
one, two, . . . , nine
eleven, twelve, . . . , nineteen
ten, twenty, . . . , ninety
start
q0
q1
q2
dollars
q4
q5
q6
cents
q7
twenty, thirty, forty, . . . ninety
one, two, three, . . . nine
twenty, thirty, forty, . . . ninety
one, two, three, . . . nine

Problem: Write an FSA for time-of-day expressions like *eleven o’clock*, *twelve-thirty*, *midnight*, *a quarter to ten* and others -- ideally, your FSA should cover all possible time-of-day expressions used in English, and only them, i.e. your FSA should not accept any expression that is ungrammatical in English.

Answer:

There are many possible solutions to this problem, and of course depending on which expressions you want your automaton to accept, the solutions can differ. Below is a reasonable solution to this problem.



$q0$
a
quarter
quarter
half
to till past
past
noon midnight
one two . . . twelve
one two . . . twelve
clock
am pm
twenty thirty . . . fifty
o'
ten eleven . . . nineteen
one two . . . nine
one two . . . nine
am pm
am pm