PROBLEM OF THE MONTH POM 2020 - 2021 **December, 2020**

MATHEMATICS

5 points: A circle is inscribed into a circular sector that has a central angle of 60 degrees. Find the ratio of the area of the inscribed circle to the area of the sector.

Hint: Use the theorem of the side of a right angle triangle opposite to the angle of 30 degrees.

Answer: 2/3

Solution: Connect the center of the inscribed circle to the vertex of the sector. Also, drop the perpendicular from the center of the circle to the side of the sector. The obtained right angle triangle is the right angle triangle with the hypotenuse $R-r$ and the side opposite to the 30-degree angle equal r, where R is the radius of the sector and r is the radius of the inscribed circle. We have $\frac{r}{R-r} = \frac{1}{2}$ and, therefore, $r = \frac{R}{3}$. Now we can easily find the ratio of the areas $\frac{\pi r^2}{\pi R^2/6} = 6(r/R)^2 = 6(1/3)^2 = 2/3$.

10 points: In triangle ABC angle A is 120 degrees, point D lies on the bisector of angle A and AD = AB + AC. Find the angles of the triangle DBC**.**

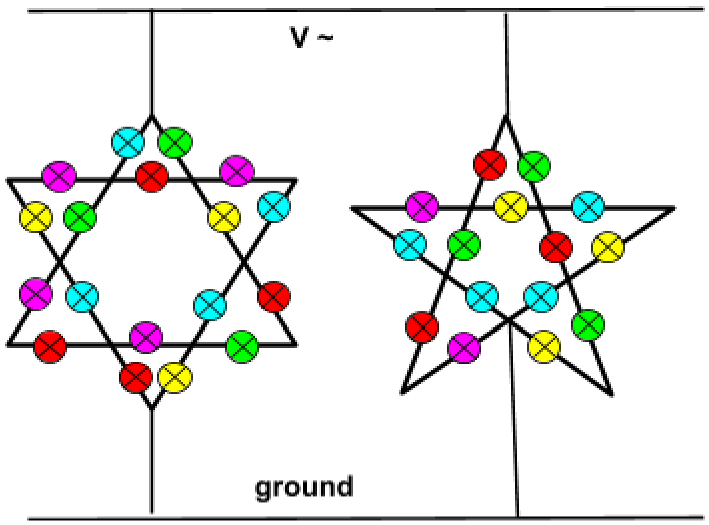
Hint: Construct equilateral triangles ABB' and ACC' with points B' and C' lying on AD. Consider the triangles BAC, CDC', and DBB'.

Answer: All angles are 60 degrees.

Solution: Construct equilateral triangles ABB' and ACC' with points B' and C' lying on AD. Consider the triangle CDC'. The angle C' of that triangle is 120 degrees as ACC' is an equilateral triangle. Also $AB' = AB$ and $AC' = AC$ by construction. Therefore, $DC' = AD - AC' = AB + AC - AC' = AB$. We conclude that the triangle $CDC' = CBA$ by SAS criterion. Therefore, $CD = BC$. Similarly, we can show that $BDB' = BCA$ and $BD = BC = CD$. We conclude that the triangle DBC is equilateral and all its angles are 60 degrees.

PHYSICS

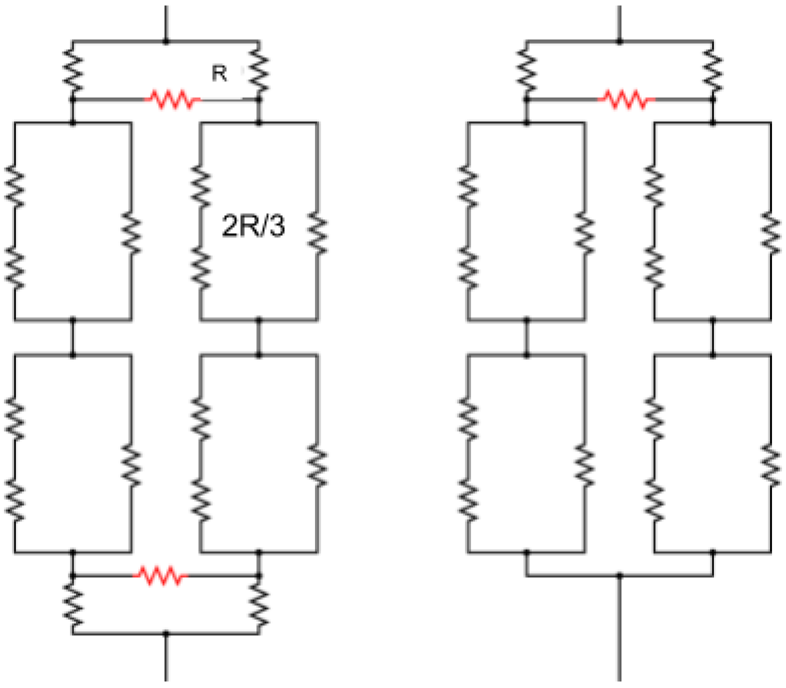
5 points: Two neighbors have built their holiday decorations: one shaped like a 5-point star and the other like a 6-point one. Each edge contains one light bulb (see the figure), all bulbs have the same constant resistance. Which of the two will consume more power when plugged into the outlet with the same voltage? Find the ratio of their respective powers. Each line represents a wire with zero resistance.



Hint: For a given voltage V, power $P = V^2/R$, where R is the total resistance of the circuit.

Answer: The 5-point star consumes more power. The power ratio is 10/7

Solution: For a given voltage V, power $P = V^2/R_{tot}$, where R_{tot} is the total resistance of the circuit. Let R be the resistance of each light bulb. Equivalent circuits for each case are shown in the figure (resistors shown in red can be removed since there is no current flowing through them). Their resistances can be calculated by using well known rules for parallel and sequential connections: R_{tot} is 5R/3 and 7R/6 for the 6-point and 5-point stars, respectively. Since the 5-point star has lower resistance, it consumes more power. The power ratio is 10/7.

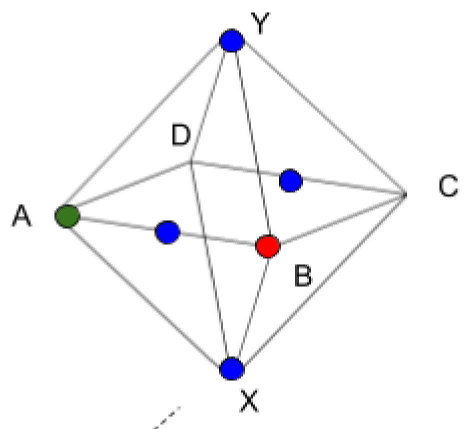


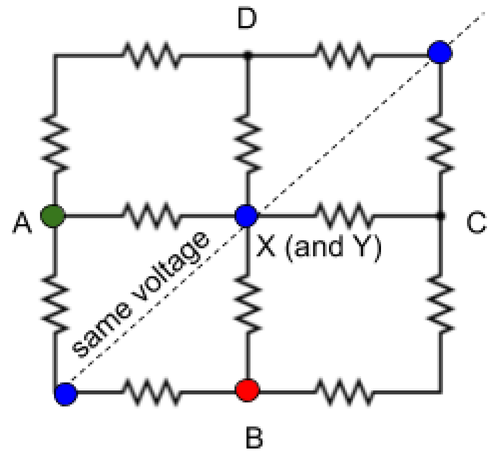
10 points: An octahedron is made out of wire. Each edge has resistance R, and each vertex is a perfect electric contact between 4 edges. Find the resistance of this circuit between two vertices that share the same edge.

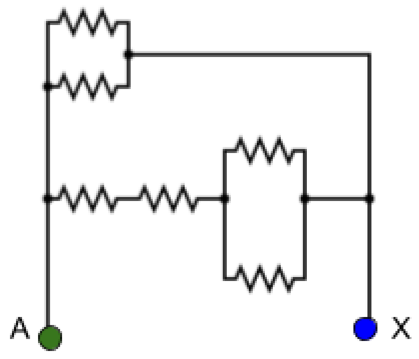
Hint: Consider a plane of symmetry half way between the two edges, A and B. Each point where it intersects the octahedron, has the same voltage (which is the average of the voltages in A and B). If we now connect all those points together, this will not change the resistance of the circuit (since no current would flow between the points at equal voltages).

Answer: 5R/12

Solution: Consider a plane of symmetry half way between the two edges, A and B. Each point where it intersects the octahedron, has the same voltage (which is the average of the voltages in A and B). If we now connect all those points together, this will not change the resistance of the circuit (since no current would flow between the points at equal voltages). These equivalent points are marked as blue in the diagram on the figure . We can redraw octahedron as an equivalent resistor circuit. Each resistor is R/2, representing either half of the edge or two edges connected in parallel (see figure). As follows from the diagram, the resistance between points A and X correspond to two resistors connected in parallel: R/4 and 5R/4. Since this gives resistance 5R/24, and the point X is halfway between A and B, the final result is 5R/12.







CHEMISTRY

This month, the topic is: **Redox reactions**
IMPORTANT! In this PoM season, we do an experiment: each month, an online lecture will be given. This lecture may be helpful for those who want to solve Chemistry PoMs, although it is not supposed to provide direct hints.
This month, the lecture will be on Dec 20 morning. At 11:00, a Zoom conference will start where October PoM solutions will be discussed. After that, approximately at 11:30, the lecture starts.
To join the Zoom conference, use this link:
https://us02web.zoom.us/j/4817690592?pwd=T2djSjRETEpDSHFZdWJpYlBTYzdjQT09
Meeting ID: 481 769 0592
Passcode: 879615

5 points:

Having lost, by the will of the elements, your backpack with food and matches on a hike to remote places, you remembered that an abandoned hut was indicated on the map nearby. Fortunately, it was easy to find. There were logs in the fireplace and cereals and salt in the cupboard, but there was no indication of any matches or lighters. During a search, an old first-aid kit was found, which contained some pills, jars of ointments, potassium permanganate, glycerol, bandages and cotton wool. In the shed, you found some rusty piece of iron, a bag of bleach and a bottle of battery acid. "Not bad, that may be a solution" - you concluded. In a few minutes you were already enjoying the warmth of the fire in the stove, on which a kettle was gurgling. How have you managed to kindle a fire?

Hint:

A fire may result if a strong solid oxidant comes into contact with reducing agents.

Answer:

The simplest way to obtain a fire would be to mix some strong oxidizer with a substance that can be easily oxidized. Usually, an oxidizer is a substance where some electronegative element is present in a moderate to high oxidation state. Gaseous oxygen is an example: oxygen is the second most electronegative element (after fluorine), and it has the oxidation state of zero in O_2 (which is high as compared to its oxidation state in, e.g. water). Another common type of oxidizers are transition metal compounds where those metals are present in their highest oxidation state. Potassium permanganate is a typical example of such an oxidizer. This compound was a very popular local disinfector in the past, and it is not a surprise that you found it in that old hut. Is formula in $KMnO_4$, and from that you can easily find that manganese oxidation state is +7, which is a maximal possible value for manganese.Therefore, you have a very nice oxidizer, and to start a fire you need just to find something that can be oxidized by potassium permanganate. The best candidate is supposed to be some liquid, because it makes a dense contact with the solid oxidizer, so the local concentration of both reactants is very high, and the reaction may go very rapidly. Obviously, glycerol would be the best choice. Chemically, it is very similar to a common sugar, so each carbon is connected just to one electronegative atom (oxygen), which means each carbon can donate up to three electrons to an oxidizer. To start a

fire, you may do something similar to various videos that you can find on youtube, just type “glycerol and permanganate”. Happy watching, and we hope that if you will find some old abandoned hut in a real life, you will probably be able to start a fire :).
Theoretically, there are some other ways tp start a fire using the provided set of materials, but the above described method is the most convenient.

10 points:

The teacher decided to answer the question of an inquisitive student about the origin of the name of one of the elements with a small demonstration. He took a bright yellow liquid and added a colorless one to it. The resulting mixture immediately turned red-orange. Then the teacher dipped a roll of aluminum foil into the resulting solution. Gas bubbles began to rise from the metal surface, and the color of the solution began to change. This happened gradually, the color changed to greenish, and over time became sky blue. The student exclaimed: "Oh, now I understand it!"
What element was the student asking about? What chemical transformations took place during these color transitions? Write down their chemical equations.

Hint:

Some transition metals have different colors in different oxidation states.

Answer:

The element’s name is Chromium. The Greec word “chroma” (χρῶμα) means “color”, and this element was named “Chromium” because its compounds have different colors depending on chromium’s oxidation state.
Thus, a salt called “potassium chromate” (K_2CrO_4) has a yellow color. It is easy to calculate that a chromium atom has an oxidation state +6 (similar to sulfur in potassium sulfate, K_2SO_4).
One interesting property of chromates is that they can dimerize in a presence of an acid and lose one molecule of water::

$$K_2CrO_4 + H_2SO_4 \rightarrow K_2Cr_2O_7 + K_2SO_4 + H_2O \qquad (1)$$

Yellow red-orange

This new salt is called potassium dichromate ($K_2Cr_2O_7$), and “di-” means it is a dimer of two chriomate molecules. This new salt has a beautiful red-orange color. Therefore, most likely, the colorless liquid the teacher added to chromate was some acid, for example, sulfuric acid, and the reaction occurred according to the equation (1).
It is easy to calculate that the oxidation state of chromium in potassium dichromate is still +6. In general, as a rule, reactions that lead to elimination of a water molecule do not lead to a change

of oxidation states of any atoms (try to think about that by yourself, if you don’t understand why, as me during the January SigmaLecture).

Besides the +6 oxidation state, chromium can exist in oxidation states +3 and +2. Usually, in the +3 state, chromium is emerald green, and in the +2 state it is cyan. To change the oxidation state, some reducing agent is needed, and the teacher decided to use a metal foil (aluminium foil would work fine). When aluminium, or zinc, or iron come to a contact with an acid (remember, the teacher added an acid to the chromate solution), the metal starts to react, and a hydrogen gas forms (below, zinc is used as an example):

$$Zn + H_2SO_4 \rightarrow ZnSO_4 + H_2 \qquad (2)$$

In reality, the reaction (2) is a two step process: during the first step, hydrogen is formed as individual atoms, and at the second step, two atoms combine together to form H_2 molecules. That fact is important, because separate atoms of hydrogen are extremely chemically active, and they quickly reduce the dichromate. Initially, trivalent chromium compounds (Cr^{+3}) are formed:

$$K_2Cr_2O_7 + 6\,H + 4\,H_2SO_4 \rightarrow K_2SO_4 + \underset{\text{(emerald-blue)}}{Cr_2(SO_4)_3} + 7\,H_2O \qquad (3)$$

and the solution is gradually becoming emerald-blue. If the reaction continues, further reduction is possible to the chromium salt where it has an oxidation state of +2, and is blue:

$$Cr_2(SO_4)_3 + 2\,H \rightarrow \underset{\text{(cyan)}}{2\,CrSO_4} + H_2SO_4 \qquad (4)$$

but it may require significant time and protection from atmospheric oxygen.

In general, chromium is a very beautiful element, although majority of its compounds are toxic or carcinogenic.

BIOLOGY

5 points:

In one of the Harry Potter books there was a creature called basilisk (a fire-spitting animal with a dragon body, rooster head, and tail of a snake). According to the Care of Magical Creatures class textbook baby-basilisk comes out of an egg, which is produced by a rooster once in 100 years and hatched by a toad. There are no other ways for reproduction.

Envision a biologically-correct life cycle that fits this description and explains why rooster and toad stages are necessary for basilisk reproduction.

Answer:

Since the problem discusses a magic creature, which does not exist in reality, the solution is not supposed to be unique. There are a lot of ways to phrase the answer, but the answer is expected to be biologically plausible. The most plausible scenarios are: (i) basilisk is a very rare (one in 100 years) mutation of a chicken that produces the resulting phenotype. Birds are very close to reptiles, so the mutation reverses the ontogenesis and leaves the individual in a form of a rooster-lizard-snake. The toad is a necessary step facilitating the hatching, either by keeping certain temperature or producing particular biologicallyt active compounds needed for a normal development of a basilisk.
The scenario (ii) is as follows: basilisk is a parasite with an extremely long life cycle, so the larva is transferred through many generations of chicken and eventually exits from a rooster in a form of an egg-like sack once in a 100 years. The next maturation step of a sack then requires certain compounds/molecules from a toad to finalise the metamorphosis.
Alteration of this is that toad is not necessary *per se*: it is only necessary that the egg should be placed in a damp and dark place, so the toad is a coincidence.
Other explanations are also possible, and if you propose something more interesting (but plausible from the point of view of modern biological science), that would be good.

10 points:

There is a specific condition called "Compound V deficiency" in humans. People suffering from this condition were treated in one of the hospitals. The first cohort of patients with this deficiency was given Compound V-rich food. Some of the patients got better, but the conditions of others did not change. Doctors decided to give them Compound V as intravenous injections. This resulted again in the full recovery of some patients, but not all of them.

Could you suggest as many reasons as you can to explain this? Also, how can the rest of the uncured patients be treated?

Answer:

Three major factors that may affect "Compound V deficiency" are (1) bioavailability - a different assimilation rate of the compound V depending on the route of administration (oral or intravenous); (2) environmental factors – some people cannot use the CompV, because they are affected by something that prevents it (for example, they have an infection or their stomach acidity is high); (3) genetic predisposition for both steps (oral or intravenous).
The most obvious is an example of iron metabolism errors – when it is given orally, it goes through the gastrointestinal system out in some people, without entering the bloodstream. The Intravenous injection can be used by almost everybody, except by people with iron-metabolism mutations in hephaestin, frataxin, etc.
Since the route of administration directly affected only the group (1) factor (bioavailability), it is not a surprise that some patients, whose "Compound V deficiency" was caused primarily by factors (2) or (3), or a combination thereof, reacted differently on the change in the route of administration of the compound V.

LINGUISTICS

5 points:

The following sequence of sounds represents a phrase in Japanese language without spaces: *kakikuebakaneganarunari*. The part of the Japanese dictionary with all words that could be present in this phrase is given below.

aka	i	na
akane	iku	nari
aki	ikue	naru
an		naruna
ana	ka	narunari
ane	kaki	ne
ari	kan	
	kane	u
baka	kanega	un
bakan	ki	unari
bakane	kiku	ueba
	kikue	
gan	ku	e
ganar	kue	eba
ganaru	kueba	
ganaruna		ri

Determine how many ways are there to break this phrase into words using this dictionary and **justify your answer**.

Answer:

216

Solution:

Looking at *ga* in the middle, it can only be the last part of *kanega* or the beginning of *gan*, *ganar*, *ganaru*, *ganaruna*.

If it's a part of kanega, then we get 20 options

Before *kanega*:

(ka-ki/kaki)-(ku-eba/kueba); ka-kiku-eba (2x2+1=5)

After *kanega*:

naru-na-ri; naru-nari; naruna-ri; narunari (4)

In the second case, if *ga* is the beginning of a word, ba can be the beginning of the end:

If ba is the end: 10 options for the first part of the word
((ka-ki/kaki)-(ku-eba/kueba); ka-kiku-eba)-(ka-ne/kane) (5x2=10)
If ba is the beginning: 18 options for the first part of the word
((ka-ki/kaki)-(ku-e/kue); ka-(kiku-e/kikue))-(bakane/baka-ne/bakan-e) (6x3=18)

Now, the second part, starting with *ga*, has 7 options:
(ganaru/ganar-u)-(nari/nar-i); ganar-(un-ari/unari); ganaruna-ri

So, in total we have: 20+(10+18)x7=216

10 points:

The following sentences are in a Caucasian language that uses a concept called **evidentiality**, where a statement is parsed differently (its grammar changes) depending on how much information is available to the speaker of that statement. Have a look at the several examples below. (ʕ and ɕ are two consonants with no direct analogues in English, while ə is a short vowel like the 'a' in *about* or 'e' in *water*.)

Muhamad Aslan dʕaydziɕit'	Muhamad thinks that Aslan came.
Aslan Fatima dchwadziɕun	Aslan thought that Fatima fell asleep.
Fatima Zarema aqəshw ljwadzəlɕit'	Fatima thinks that Zarema washed the window.
Aslan chay yzhwədzəlɕit'	She thinks that Aslan drank the tea.
Muhamad chay lzhwədziɕun	Muhamad thought that she drank the tea.

Translate the following sentences into English and **explain your solution**. The last sentence (c) has two translations (try to find both of them):
a. Aslan Muhamad aqəshw yjwadziɕun
b. Muhamad dchwadzəlɕit'
c. Zarema dʕaydzəlɕit'

Also, translate these statements into the Caucasian language and **explain your solution**:
d. Aslan thought that Zarema drank the tea.
e. She thought that he fell asleep.

Hint:

Answer:

a. Aslan thought that Muhamad washed the window.
b. She thinks that Muhamad fell asleep.
c. She thinks that Zarema came **AND** Zarema thinks that he/she came
d. Aslan Zarema chay lzhwədzəlɕit'
e. Dchwadzəlɕun

Solution:

dz-V(I)-ʁ → indicate **inferential** possibility (the speaker **thinks** that something happened)
-dziʁ- appears for male speakers, -dzəlʁ- appears for female speakers
ENDINGS: -it’ for present tense, -un for past tense
INITIALS: d- for intransitive verbs (no object), l- / y- for transitive verbs (with object; l- is for female subjects, y- is for male subjects)
The speaker and subject can be absent from the sentence, as they are accounted for in the structure of the verb.
VERB ROOTS / OBJECTS:

-ʕay-	to have come
-chwa-	to have fallen asleep
-jwa-	to have washed
-zhwə-	to have drank
aqəshw	window
chay	tea

a. Aslan thought that Muhamad washed the window.
b. She thinks that Muhamad fell asleep.
c. She thinks that Zarema came **AND** Zarema thinks that he/she came
d. Aslan Zarema chay lzhwədzəlʁit’
e. Dchwadzəlʁun

COMPUTER SCIENCE

- Your program should be written in Java or Python-3
- No GUI should be used in your program: eg., easy gui in Python
- All the input and output should be via files with specified in the problem statement
- Java programs should be submitted in a file with extension .java; Python-3 programs should be submitted in a file with extension .py.
  No .txt, .dat, .pdf, .doc, .docx, etc. Programs submitted in incorrect format will not receive any points!

Introduction:

This month the problem will deal with word shuffles. We will define a word shuffle as a combination of two words where the letters of the shuffle come from the original words in such a way that the relative order of the letters coming from the same word is maintained, however each letter of the shuffle can be drawn from either of the words. For example, a shuffle of TOURNAMENT and DINNER could be TDINOURNANMENTER. The shuffle would essentially weave two original words together utilizing all the letters.

5 points:

Write a program that given two original words and a shuffle would determine whether the shuffle is "legitimate" (i.e. produced correctly). The input of your program is a file **input.txt** , which contains 3 lines: two original words and the shuffle. The output of your program is a file **output.txt**, which should contain either CORRECT, if the shuffle is produced correctly from two original words, or INCORRECT otherwise.

Solution:

Java:

```
/*
We will scan the words sequentially and check while it matches. If we got the same letter from
2 words, i.e. either of them can match, we'll check both variants.
*/

import java.io.*;

public class WordShuffle5 {
  public String word1;
  public String word2;
  public String shuffle;

  void input(String fname) throws IOException {
    try(BufferedReader br = new BufferedReader(new FileReader(fname))) {
      word1 = br.readLine().trim().toUpperCase();
      word2 = br.readLine().trim().toUpperCase();
      shuffle = br.readLine().trim().toUpperCase();
      if(word1.isEmpty() || word2.isEmpty() ||
shuffle.length()!=word1.length()+word2.length())
        throw new IllegalArgumentException("invalid input");
    }
  }

  void output(String fname, boolean isCorrect) throws IOException {
    try(BufferedWriter bw = new BufferedWriter(new FileWriter(fname))) {
      bw.write(String.format("%s\n", isCorrect ? "CORRECT" : "INCORRECT"));
    }
  }

  boolean checkShuffle(String word1, String word2, String shuffle) {
    char ch1 = word1.charAt(0);
    String tail1 = word1.substring(1);
    if(ch1 == shuffle.charAt(0)) {
      if(tail1.length() > 0) {
        if(checkShuffle(tail1, word2, shuffle.substring(1)))
          return true;
      }
      else { // the 1st word is exhausted
        if(word2.isEmpty() || word2.equals(shuffle.substring(1)))
          return true;
      }
    }
    char ch2 = word2.charAt(0);
    String tail2 = word2.substring(1);
    if(ch2 == shuffle.charAt(0)) {
      if(tail2.length() > 0) {
        if(checkShuffle(word1, tail2, shuffle.substring(1)))
          return true;
      }
      else { // the 2nd word is exhausted
        if(word1.isEmpty() || word1.equals(shuffle.substring(1)))
          return true;
      }
    }
    return false;
  }

  public static void main(String[] args) throws Exception {
    WordShuffle5 ws = new WordShuffle5();
    ws.input("input.txt");
    boolean isCorrect = ws.checkShuffle(ws.word1, ws.word2, ws.shuffle);
    ws.output("output.txt", isCorrect);
    System.out.println("end.");
  }
}
```

Python-3:

```
"""
We will scan the words sequentially and check while it matches. If we got the same letter from
2
words, i.e. either of them can match, we'll check both variants.
"""
```

```
# read and parse input file
def input(fname):
  with open(fname) as in_file:
    word1 = in_file.readline().strip()
    word2 = in_file.readline().strip()
    shuffle = in_file.readline().strip()
    assert(len(word1)>0 and len(word2)>0 and len(shuffle)==len(word1)+len(word2))
    return word1.upper(), word2.upper(), shuffle.upper()

# output answer
def output(fname: str, is_correct: bool) -> None:
  with open(fname, "w") as out_file:
    out_file.writelines(f"{'CORRECT' if is_correct else 'INCORRECT'}\n")

def check_shuffle(word1, word2, shuffle):
  ch1 = word1[0]
  tail1 = word1[1:]
  if ch1 == shuffle[0]:
    if len(tail1) > 0:
      if check_shuffle(tail1, word2, shuffle[1:]):
        return True
    else: # the 1st word is exhausted
      if len(word2)==0 or word2==shuffle[1:]:
        return True
  ch2 = word2[0]
  tail2 = word2[1:]
  if ch2 == shuffle[0]:
    if len(tail2) > 0:
      if check_shuffle(word1, tail2, shuffle[1:]):
        return True
    else: # the 2nd word is exhausted
      if len(word1)==0 or word1==shuffle[1:]:
        return True
  return False

word1, word2, shuffle = input("input.txt")
is_correct = check_shuffle(word1, word2, shuffle)
output("output.txt", is_correct)
print(f"is_correct = {is_correct}")
print("end.")
```

10 points:

Write a program that given a shuffle would determine which two words from the list of two English words were used to produce it. The list of allowed words is supplied via words.txt file (you can download this file here: https://raw.githubusercontent.com/eneko/data-repository/master/data/words.txt).

Input is provided in **input.txt** file which contains a single word - the shuffle.

Output should be written to **output.txt** file and should contain two original words in alphabetical order, or, if the provided shuffle cannot be produced from any combination of the provided words, write IMPOSSIBLE. If there are multiple pairs of words that produce the input shuffle, then write any one of the pairs to the output.txt file.

Solution:

Java:

```
/*
We'll use a similar (but not exact) idea as in the 5-pointer. We'll scan the shuffle
sequentially
and see if the current letter belongs to the first or second word. To simplify the word
matching
we'll organize our dictionary into a Trie (see https://en.wikipedia.org/wiki/Trie ).
*/

import java.io.*;
import java.util.HashSet;
import java.util.Set;

class Trie {
  static final char FIRST_LETTER = '-';
  static final char LAST_LETTER  = '_';
  static final int ALPHABET_SIZE = LAST_LETTER - FIRST_LETTER;

  static class TrieNode {
    TrieNode[] children;
    boolean isEndOfWord;

    TrieNode() {
      children = new TrieNode[ALPHABET_SIZE];
      for(int i=0; i<ALPHABET_SIZE; i++)
        children[i] = null;
      isEndOfWord = false;
    }
  }

  TrieNode root;

  Trie() {
    root = new TrieNode();
  }

  // If not present, inserts a word into trie. If the word is a prefix of a trie node,
  // just marks leaf node.
  void insert(String word) throws Exception {
    TrieNode cur = root;
    for(char letter : word.toCharArray()) {
      int index = letter - FIRST_LETTER;
      if(index<0 || index>=ALPHABET_SIZE)
        throw new Exception("bug: check FIRST_LETTER and LAST_LETTER");
      if(cur.children[index] == null)
        cur.children[index] = new TrieNode();
      cur = cur.children[index];
    }
    cur.isEndOfWord = true;
  }
  // Returns true if key presents in trie, else false.
  boolean search(String word) {
    TrieNode cur = root;
    for(char letter : word.toCharArray()) {
      int index = letter - FIRST_LETTER;
      if(cur.children[index] == null)
        return false;
      cur = cur.children[index];
    }
    return(cur != null && cur.isEndOfWord);
  }
}

public class WordShuffle10 {
  public String word1;
  public String word2;
  public String shuffle;
  public Set<String> dict = new HashSet<>();
  public int count = 0;

  static class WordsMatch {
    public boolean isFirst;
    public String  word1;
    public boolean isSecond;
    public String  word2;

    public WordsMatch(boolean isFirst, String word1, boolean isSecond, String word2) {
      this.isFirst  = isFirst;
      this.word1    = word1;
      this.isSecond = isSecond;
      this.word2    = word2;
    }
  }

  void input(String fname, String dname) throws IOException {
    try(BufferedReader br = new BufferedReader(new FileReader(fname))) {
      shuffle = br.readLine().trim().toUpperCase();
      if(shuffle.length() < 2)
        throw new IllegalArgumentException("invalid input");

      try(BufferedReader br2 = new BufferedReader(new FileReader(dname))) {
        String word;
        while((word=br2.readLine()) != null) {
          dict.add(word.trim().toUpperCase());
        }
      }
    }
  }

  void output(String fname, String word1, String word2) throws IOException {
    try(BufferedWriter bw = new BufferedWriter(new FileWriter(fname))) {
      if(word1==null || word1.isBlank())
        bw.write("IMPOSSIBLE\n");
      else if(word1.compareTo(word2) < 0)
        bw.write(String.format("%s\n%s\n", word1, word2));
      else
```

```
        bw.write(String.format("%s\n%s\n", word2, word1));
      }
    }

    Trie makeTrie(Set<String> words) throws Exception {
      Trie trie = new Trie();
      for(String word : words) {
        trie.insert(word);
      }
      return trie;
    }

    WordsMatch checkShuffle(String word1, Trie.TrieNode trie1, String word2, Trie.TrieNode
trie2,
                            String shuffle) {
      if(shuffle.isEmpty()) // we've examined all letters in shuffle
        return new WordsMatch(trie1.isEndOfWord, word1, trie2.isEndOfWord, word2);
      if(trie1.children[shuffle.charAt(0)-Trie.FIRST_LETTER] != null) { // a word exists with
such
                                                                         // prefix
        count++;
        WordsMatch wordsMatchNew = checkShuffle(word1+shuffle.charAt(0),

trie1.children[shuffle.charAt(0)-Trie.FIRST_LETTER],
                                              word2, trie2, shuffle.substring(1));
        if(wordsMatchNew.isFirst && wordsMatchNew.isSecond) // found a match for both words
          return new WordsMatch(wordsMatchNew.isFirst, wordsMatchNew.word1,
wordsMatchNew.isSecond,
                                wordsMatchNew.word2);
      }
      if(trie2.children[shuffle.charAt(0)-Trie.FIRST_LETTER] != null) { // a word exists with
such
                                                                         // prefix
        count++;
        WordsMatch wordsMatchNew = checkShuffle(word1, trie1, word2+shuffle.charAt(0),

trie2.children[shuffle.charAt(0)-Trie.FIRST_LETTER],
                                              shuffle.substring(1));
        if(wordsMatchNew.isFirst && wordsMatchNew.isSecond) // found a match for both words
          return new WordsMatch(wordsMatchNew.isFirst, wordsMatchNew.word1,
wordsMatchNew.isSecond,
                                wordsMatchNew.word2);
      }
      return new WordsMatch(false, word1, false, word2);
    }

    WordsMatch crackShuffle() throws Exception {
      Trie trie = makeTrie(dict);
      return checkShuffle("", trie.root, "", trie.root, shuffle);
    }

    public static void main(String[] args) throws Exception {
      // System.out.println("pwd = " + System.getProperty("user.dir"));
      WordShuffle10 ws = new WordShuffle10();
      ws.input("input.txt", "words.txt");
      System.out.printf("there are %,d words in our dictionary\n", ws.dict.size());
      WordsMatch wordsMatch = ws.crackShuffle();
      boolean found = wordsMatch.isFirst && wordsMatch.isSecond;
      System.out.printf("found: %b, %s, %s; count = %d\n", found, wordsMatch.word1,
                        wordsMatch.word2, ws.count);
      ws.output("output.txt", wordsMatch.word1, wordsMatch.word2);
      System.out.println("end.");
    }
  }
```

Python-3:

```
"""
We'll use a similar (but not exact) idea as in the 5-pointer. We'll scan the shuffle
sequentially
and see if the current letter belongs to the first or second word. To simplify the word
matching
we'll organize our dictionary into a Trie (see https://en.wikipedia.org/wiki/Trie).
"""

# read and parse input file
def input(fname, dict):
  with open(fname) as in_file:
    shuffle = in_file.readline().strip()
    assert(len(shuffle) >= 2)
    with open(dict) as in_file:
      d = set()
      for word in in_file:
        d.add(word.strip().upper())
      return shuffle.upper(), d

# output answer
def output(fname, word1, word2):
  with open(fname, "w") as out_file:
    if not word1:
      out_file.writelines(f"IMPOSSIBLE\n")
    elif word1 <= word2:
      out_file.writelines(f"{word1}\n{word2}\n")
    else:
      out_file.writelines(f"{word2}\n{word1}\n")

def make_trie(words):
  root = {}
  for word in words:
    current_dict = root
    for letter in word:
      current_dict = current_dict.setdefault(letter, {})
    current_dict['*'] = '*' # terminal marker
  return root

count = 0

def check_shuffle(word1, trie1, word2, trie2, shuffle):
  global count
  if len(shuffle) == 0: # we've examined all letters in shuffle
    return '*' in trie1, word1, '*' in trie2, word2 # "'*' in trie1" means word1 matched to
the
```

```
                                                    # end
  if shuffle[0] in trie1: # a word exists with such prefix
    count += 1
    is_first, word1_new, is_second, word2_new = check_shuffle(word1+shuffle[0],
                                                              trie1[shuffle[0]], word2, trie2,
                                                              shuffle[1:])
    if is_first and is_second: # found a match for both words
      return is_first, word1_new, is_second, word2_new
  if shuffle[0] in trie2:
    count += 1
    is_first, word1_new, is_second, word2_new = check_shuffle(word1, trie1, word2+shuffle[0],
                                                              trie2[shuffle[0]], shuffle[1:])
    if is_first and is_second:
      return is_first, word1_new, is_second, word2_new
  return False, word1, False, word2

def crack_shuffle(shuffle, dict):
  trie = make_trie(dict)
  is_first, word1, is_second, word2 = check_shuffle(word1="", trie1=trie, word2="",
trie2=trie,
                                                    shuffle=shuffle)
  return is_first and is_second, word1, word2

shuffle, dict = input("input.txt", "words.txt")
print(f"there are {len(dict):,d} words in our dictionary")
found, word1, word2 = crack_shuffle(shuffle, dict)
print(f"found: {found}, {word1}, {word2}; count = {count}")
output("output.txt", word1, word2)
print("end.")
```