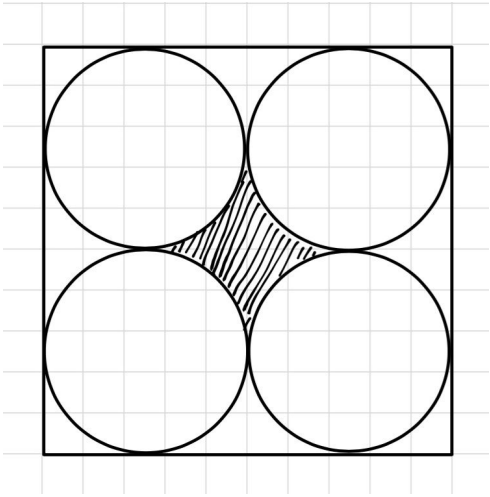
PROBLEM OF THE MONTH

October, 2016

2016-2017

MATHEMATICS

5 points: Find the area of the star-shaped region constructed from the square with side a as shown in the figure.

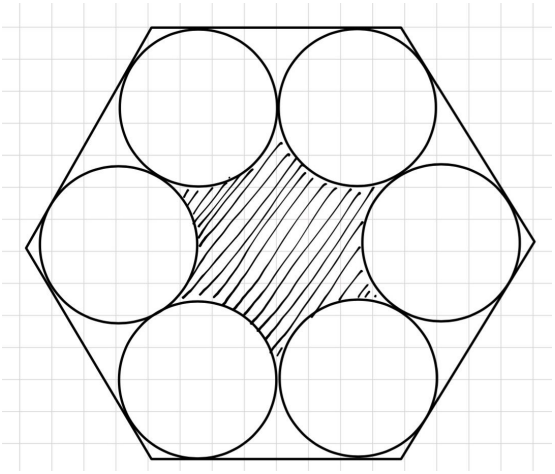


Hint: Connect the centers of circles and consider the obtained square.

Answer: $(1-\frac{\pi}{4})\frac{a^2}{4}$

Solution: If we connect the centers of circles we obtain the square with the side $a/2$. The area of this square is $\frac{a^2}{4}$. To obtain the area of the star-shaped region we should subtract from this the area of the circle of the radius $a/4$. We obtain $(1-\frac{\pi}{4})\frac{a^2}{4}$.

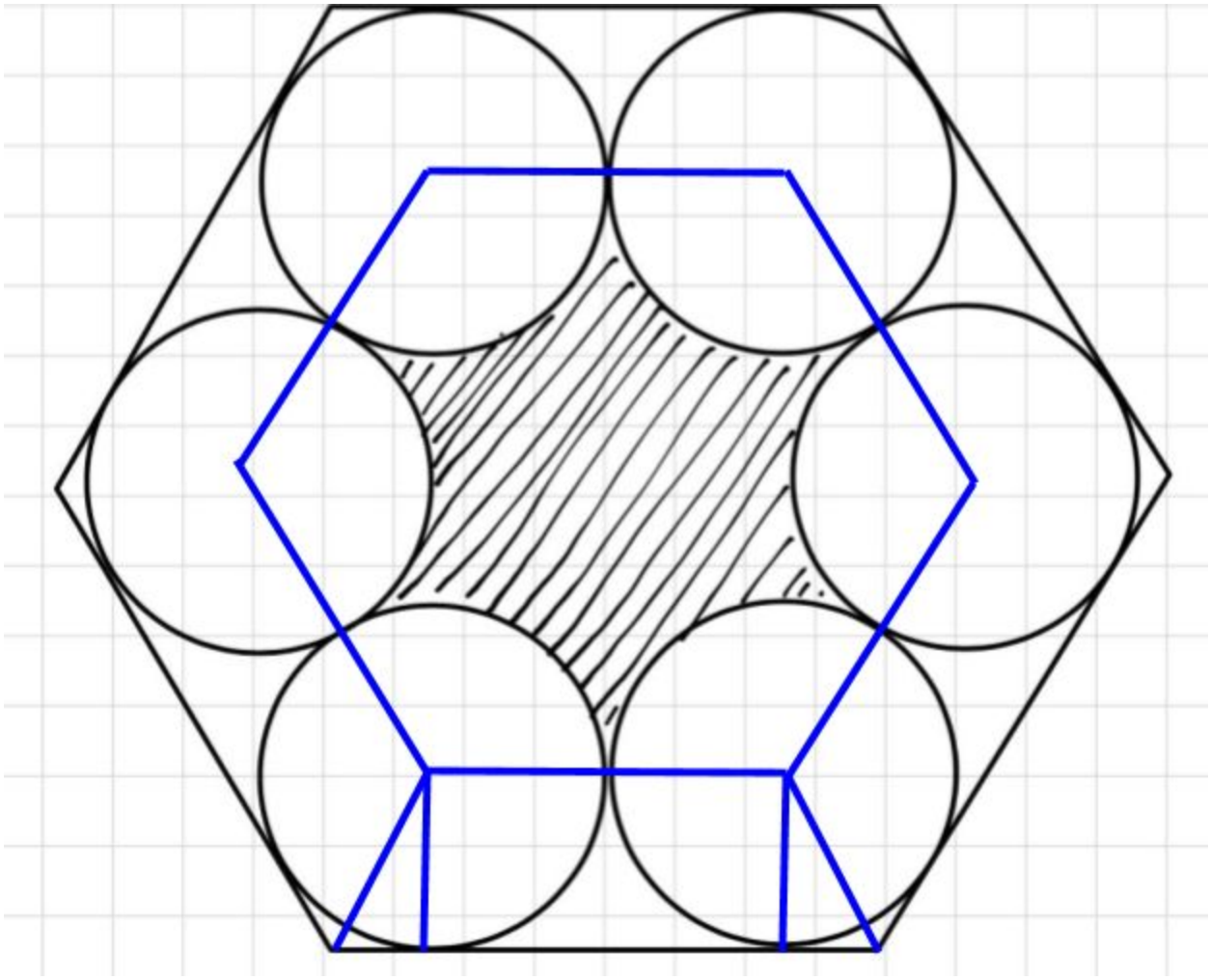
10 points: Find the area of the star-shaped region constructed from the regular hexagon with side a as shown in the figure.



Hint: Connect the centers of circles and consider the obtained hexagon.

Answer: $\frac{3}{4}\,(3\sqrt{3}-\pi)(2-\sqrt{3})a^2$

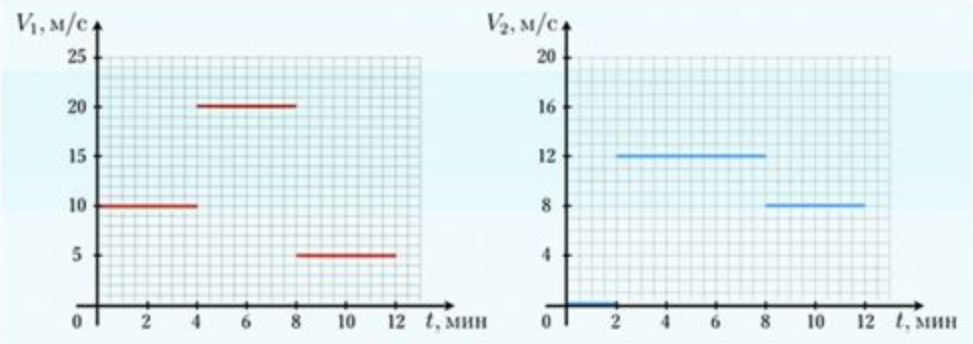
Solution: Let us connect the centers of circles. We obtain the hexagon with the side $2R$, where R is the radius of each circle. This radius can be found from the trapezoid shown in the figure as follows $a = 2(R + R/\sqrt{3})$ and then $R = \frac{1}{4}(3-\sqrt{3})a$, where we used that the smallest angle of a small triangle is 30°. The area of the hexagon formed by circle centers can be found as $6 * \frac{1}{2}\, 2R * (2R\frac{\sqrt{3}}{2}) = 6\sqrt{3}\, R^2$. The sum of the areas of all 6 sectors inside this hexagon is $6 * \frac{1}{3}\, \pi\, R^2 = 2\pi R^2$. The area of the star-shaped region is given by the difference $6\sqrt{3}\, R^2 - 2\pi R^2 = 2(3\sqrt{3}-\pi)R^2 = \frac{3}{4}\,(3\sqrt{3}-\pi)(2-\sqrt{3})a^2$.



PHYSICS

5 points:

The two graphs below represent speeds of two friends biking in opposite directions. Biker #2 started moving two min later and biked 7 minutes until he met his partner. Find the initial distance between the bikers at t=0.



Hint: Find how much each biker travelled over each time interval when the speed was constant, up to the moment *t=9 min*.

Answer: $12.3\ km$

Solution: In the first 4 minutes, the first biker covered $10\frac{m}{s} \cdot 4 \cdot 60s = 2400m$, in the next 4 minutes - another $20\frac{m}{s} \cdot 4 \cdot 60s = 4800m$, and in the final 1 minute before the meeting, $5\frac{m}{s} \cdot 1 \cdot 60s = 300m$. This gives the total distance travelled by the first biker to be $2400m + 4800m + 300m = 7500m$. The second bikers had travelled the distance $12\frac{m}{s} \cdot 6 \cdot 60s + 8\frac{m}{s} \cdot 1 \cdot 60s = 4800m$ until they met. Therefore, the initial distance separating the two friends was $7500m + 4800m = 12.3km$.

10 points:

42 freight carriages of the same mass initially stand still on rails spaced by equal gaps of 10 cm. At certain moment, the first carriage is pushed and starts moving towards the next one with initial speed $v = 36\ km/h$. When a moving carriage hits the next one, a coupling mechanism is engaged, and they get rigidly connected. This repeats for all the

subsequent collisions between the carriages. Find the time between the first and the last such collisions.

Hint: Note that at each collision momentum is conserved, but energy is not. This means that when *n* carriages are moving together, they have the same momentum as the first carriage in the very beginning. Can you find their speed? The time it will take for them to collide with the next carriage?

Answer: $8.6\ s$.

Solution: Note that at each collision momentum is conserved, but energy is not. This means that when *N* carriages are moving together, they have the same momentum as the first carriage in the very beginning. This means that their speed is *v/N*. Before colliding with the next carriage, they will move for time $t = \frac{d}{v/N} = N \cdot 0.01s$ (here $v = 36\frac{km}{hr} = 10\frac{m}{s}$ is original speed and $d = 0.1m$ is the gap).
The total time between the first and last collisions is therefore

$$T = 0.01s \cdot (2 + 3 + ...40 + 41) = 0.01s \cdot 40 \cdot 43/2 = 8.6\ s.$$

CHEMISTRY

5 points:

(Halloween problem)

We all know that will-o'-the-wisp, or *ignis fatuus,* a ghost light travellers see at night over swamps, marshes and bogs is a standard element of fairy tales and scary stories. It is generally believed *ignis fatuus* is the result of combustion of the mixture of methane with small traces of phosphine that form during anaerobic decomposition of organic materials. However, this theory seems to contradict with what we know about combustion of methane. Indeed, methane-air mixtures that contain more than 17% of methane are too methane rich to ignite: there is not enough oxygen to support combustion, so there simply will be no burn. The mixtures containing less than 4.4% of methane also cannot burn, because there is not enough methane there. However, when the methane concentration is in between 4.4% and 17%, combustion does not start spontaneously; at least a small spark is needed for that. Once such combustion has started, it continues violently, in a form of a violent explosion, and it doesn't look like *ignis fatuus.*
Please, explain (i) what initiates the combustion of the swamp gas (remember, methane-air mixture cannot ignite spontaneously), and (ii) why fire *ignis fatuus* look like faint slow light, not violent explosions.

Hint:

Try to read more about properties of phosphine, and about the phenomenon of "cold fire"

Solution:

Methane burns at open air, and this reaction produces a bright light. However, for this reaction to occur, high temperature is needed: a methane molecule is pretty stable, and oxygen cannot break a carbon-hydrogen bond at room temperature. In contrast, in a presence of a minor fire even small concentrations of methane can produce some low intensity faint fire. This fire is not self-sustained, however, it lasts as soon as the primary source of fire is present. In contrast to methane, phosphine (PH_3) reacts with oxygen even at room temperature. Phosphine/air mixtures ignite spontaneously even when phosphine's concentration is low. The amount of phosphine produced in swamps is not sufficient to cause *ignis fatuus* by itself, but since methane is being produced in much higher concentration (although still below 4.4%), self-ignition of phosphine provides a continuous source of primary fire that supports slow and low temperature combustion of methane, which is visible as *ignis fatuus*.

10 points:

In “The Terminator” film, Kyle buys and brings to the hotel the moth balls, corn syrup, and ammonia. Using this set of chemicals, is it possible to make a weapon that will destroy the Terminator? If yes, explain how. If not, explain what else should Kyle buy in the store. Limit yourself with the materials available for purchase in common stores. Of course, we don’t need to know the exact recipe, we discuss it from a purely theoretical point of view.

Hint:

To kill the Terminator, Kyle had to make some explosive (he mentioned nitroglycerol, but it is more likely he prepared something more simple and safe). The explosion (if it is not a nuclear explosion) is a liberation of energy during some chemical reaction. The most violent reactions are the reactions between a strong oxidizer and some reducing agent. A lot of materials available in stores can serve as reducing agents, for example, sugar or charcoal. However, Kyle also needs an oxidizer. What could it be?

Solution:

A chemical explosion is always a reaction between some oxidizer and some reducing agent. For example, gunpowder is a mixture of saltpeter (potassium nitrate, an oxidizer) and sulfur+charcoal (reducing agent). Accordingly, an explosion is a kind of a violent combustion reaction that requires no external oxidizer (e.g. air). Sometimes, the same molecule may serve both as an oxidizer and a reducer. For example, in nitroglycerol, three nitro groups are oxidizers that oxidize other part of the molecule (a glycerol), which lead to an explosion.

Among the materials Kyle brough to the hotel there are no oxidizers, so it is impossible to make an explosive from them. However, since we do not know what else did he buy in addition to the items listed, it is quite possible that he bought potassium nitrate (a nitrogen fertilizer) from the Home Depot, and that may allow him to prepare a simple gunpowder. In the movie, he also mentioned nitroglycerol, and theoretically he could prepare it from the nitrogen fertilizer, battery acid (which is also available eitgher in the Home Depot or from an old car battery), and glycerol (a component of some cosmetics). However, preparation of nitroglycerol is extremely dangerous, so by attempting to prepare it Kyle would just make the Terminator’s task easier (by killing himself and Sarah in the huge explosion). Therefore, it is unlikely that he decided to prepare nitroglycerol in the hotel room. What is more likely, that he bought ammonium nitrate, another nitrogen fertilizer that is commercially available. Ammonium nitrate is an explosive by itself,

because it contains both an oxidizer (nitrate) and a fuel (ammonia). However, its explosion is very hard to initiate. To do this, Kyle needed some very efficient primary explosive, such as lead aside. Just few grams of lead azide are needed for that, but it is very hard to prepare.

Another option is to use hydrogen peroxide as an oxidant and acetone as a fuel. These two compounds combine together yielding acetone peroxide, a highly explosive solid. However, this solid is very unstable, and will explode even as a result of a small shock. That means the chase scene described in the film would be impossible (the bombs make using this explosive would explode directly in the car).

In addition, although both acetone and hydrogen peroxide are possible to buy, they are unsuitable for synthesis of explosive, because hydrogen peroxide solution is too dilute.

In summary, although Kyle was theoretically capable of preparing some explosives from the materials purchased in a store, it would be virtually impossible to make a reasonably reliable and efficient explosives in the hotel room without special equipment and in a reasonable time.

BIOLOGY

5 points:

(Halloween problem) In the early 20th century, scientists attempted to provide a viable explanation/reasoning for various manifestations of vampirism. For example, actual reason behind the legend about vampire's intolerance of garlic smell was the allergic reaction of people believed to be vampires to garlic. This reaction could be severe and frequently lead to asphyxiation. Please try to give a medical/scientific explanation for other tendencies associated with vampires, such as:

- fear of daylight
- avoidance of contact with silver
- repulsion of rooster call
- tendency to sleep in burial vaults and tombs
- desire to consume fresh blood
- very pale skin
- avoidance of contact with aspen wood

Answer:

A disease called pellagra that is currently rare and mainly dietary in origin results in many of symptoms colloquially associated with vampires and other blood-thirsty creatures. It is individuals suffering from pellagra that are sometimes speculated to have served as the inspiration of mythical stories. Pellagra's onset is a result of vitamin B3 (niacin) deficiency, and it is usually preceded by a prolonged excessive reliance on consumption of corn or corn derived food (which contains no niacin), and very little of everything else. Pellagra manifests itself as a set of behavioral and physiological changes that in some aspects look "vampirish".

Behaviorally, pellagra's symptoms are anxiety, insomnia and even aggression, which fit common stereotypes of vampire's behavior.

The early physiological changes associated with pellagra are related mostly to development of severe glossy rashes with skin areas affected by the rashes eventually settling in brown coloration. These changes are exacerbated by exposure to sunlight, which served as a ground for the development of the myth about perceived vampires' **fear of daylight**, which in turn transformed at a later period into the belief that vampires have a **tendency to sleep in burial vaults and tomb**, as well as into the myth about **repulsion of rooster call** (which has been speculated to be a psychological association to sunrise which leads to more sunlight exposure).

Pale skin of pellagra patients is an indirect consequence of skin rashes and sunlight intolerance, and appear when sunlight is avoided for long periods.
Another physiological consequence of pellagra is its frequent association with stomach bleeding and difficulty digesting normal food, which may have been misinterpreted as "digesting nothing but blood", and later transformed into the myth about vampires' d**esire to consume fresh blood.**

With regard to other traits of vampirism, **such as avoidance of contact with silver** or with **aspen wood,** they have no solid rational explanations, although the origin of the myth about avoidance of silver may be traced back to the unsuccessful attempts to treat pellagra with silver salts (e.g. silver nitrate). Since silver is also an alchemical symbol of the Moon, it is possible that early physicians, who saw a causal linkage between pellagra patient's fear of light and the Moon, decided to use silver for treatment of this disease, which, unfortunately, lead to severe consequences.

10 points:

In 2011, health officials noticed that children in Germany, Austria, and Switzerland who grow up drinking raw milk don't get asthma and other allergies (e.g., hay fever). If you wanted to figure out why, how would you design the best experiment that you can think of in order to test what mechanism is responsible? Be careful that your design provides a way to rule out other factors that have nothing to do with the milk. Link your experiment to specific hypotheses.

Hint: Since the commercial milk is assayed and pronounced "clean" in order to be sold, it must be that it's not so much that the commercial milk has something bad, but rather that the raw milk has something good/protective that is being killed off by pasteurization. Control conditions need to eliminate other factors that have nothing to do with the milk; for example, we would need to control for children who grow up on farms versus other (less healthy) environments.

Answer: There is more than one way to set up an experiment, and any that is logical and controls for confounds will receive full points. Here is one example of a potential solution:

1.Set up an experiment in which there are three groups of children: **RMilk**: those who are drinking raw milk; **PMilk**: those who are drinking pasteurized milk, and **NMilk**: those who are drinking no milk. Make sure that all three groups have equal numbers of boys and girls, equal numbers of city-dwellers and country/farm-dwellers, and equal

income, education, and access to medical care. Then, see if all three groups have equal rates of asthma.

2.If yes, then we know that the “raw milk” effect was due to something other than raw milk: either gender, living environment, or access to medical care. You can then conduct experiments on each variable (boys vs. girls, city vs. country, etc.) to identify the variable that makes a difference in terms of number of people with asthma.

3.If no, then we can have more confidence that there’s something in the milk itself that is affecting asthma.

4.If so, then we then want to figure out whether the raw milk has a component that is protective, but which is being killed off by pasteurization, versus whether the pasteurized milk has a component that is bad, which is being avoided in raw milk. To do this, we can do experiments using raw milk, pasteurized milk, and no milk, in order to determine their influence on bacteria, viruses, and human immune cells.

5.If pasteurized milk has a different effect on bacteria, viruses, or human immune cells than both raw milk and no milk, then we know that raw milk prevents asthma by avoiding something bad in pasteurized milk (which we can then do further experiments to identify…).

6.If, on the other hand, raw milk has a different effect on bacteria, viruses, or human immune cells than both pasteurized milk and no milk, then we know that raw milk prevents asthma by adding something good that is killed off in pasteurization (which we can then do further experiments to identify…).

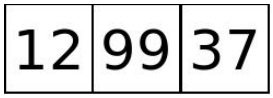
COMPUTER SCIENCE

- 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!

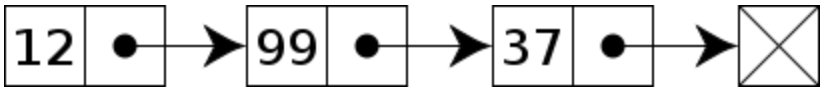
Introduction: Lists

(If you already know what single-dimensional and multidimensional arrays are, you are welcome to skip straight to the problems below)

You are already familiar with a concept of an array - a data structure used by computer languages to store a sequence of values. **List**, or more formally, **linked list**, is another data structure used to store a sequence of values. If in an array all the values are stored in array elements located next to each other in *contiguous* area of memory, like this:



lists consist of a collection of nodes, with each node containing a single value plus a *link* to the next node in sequence:



A linked list whose nodes contain two fields: an integer value and a link to the next node. The last node is linked to a terminator used to signify the end of the list.

Nodes of a linked list do not need to be in contiguous area of memory. The size of the arrays is fixed, so we must know the upper limit on the number of elements in advance. Also, generally, the allocated memory is equal to the upper limit irrespective of the

usage. On opposite, the size of the lists is not fixed - lists can grow or shrink dynamically as the new nodes are deleted or removed.

Inserting a new element in an array of elements is expensive, because room has to be created for the new elements, and to create room existing elements have to shifted. For example, suppose we maintain a sorted list of IDs in an array ids[].

ids[] = [1000, 1010, 1050, 2000, 2040, …..]

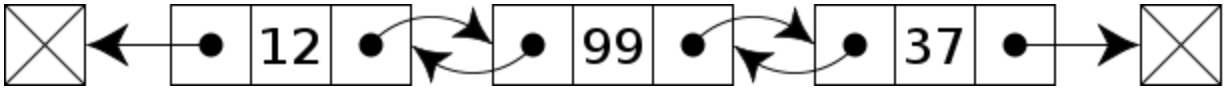
And if we want to insert a new ID 1005, then to maintain the sorted order, we have to move all the elements after 1000 (excluding 1000). Deletion is also expensive with arrays. For example, to delete 1010 in ids[], everything after 1010 has to be moved.

With lists, and insertion (or deletion) is simple: a space for a new node needs to be allocated, and then to insert the new node in the list all what needs to be done is to adjust the links around the area of insertion.

However, compared with arrays, lists have the following drawbacks:

- We can't simply access N-th node of a list as we access N-th element of an array. We have to access elements sequentially starting from the first node.
- Extra memory space is required in each node of the list to store a link.

By the way, list nodes may contain not one, but two links: one to the previous node of the list and one to the next node. Such a list is called doubly linked list:



A doubly linked list whose nodes contain three fields: an integer value, the link forward to the next node, and the link backward to the previous node

Different languages have different syntax for lists. Look up the list documentation for the language of your choice to learn how to implement them in your code.

5 points:

Card players traditionally "cut" the deck before each game: after the cards are shuffled one of the players is offered to take an arbitrary number of cards from the top of the deck and place them at the bottom. We will emulate this process in our assignment. For simplicity our cards will contain integer numbers. Your program should enter N cards

constituting a deck. Then it should ask the user how many cards (M) he or she would like to cut. In response, first M cards should be moved to the end.
For example, if original deck was [1,2,3,4,5,6,7,8] after cutting 3 cards resulting deck is [4,5,6,7,8,1,2,3]. Your program should print the resulting deck. Use a list to implement the procedure.
Additionally, please calculate how many how many swap or move operations would be required if you would have implemented your program using an array.

Solution:

First, let's start with the last question: "calculate how many how many swap or move operations would be required if you would have implemented your program using an array."

We can use N swaps (a pair exchange) to rearrange an array. For our example, we could use the following steps (this is not the only solution):
1,2,3,4,5,6,7,8
6,2,3,4,5,1,7,8
6,7,3,4,5,1,2,8
6,7,8,4,5,1,2,3
4,7,8,6,5,1,2,3
4,5,8,6,7,1,2,3
4,5,7,6,8,1,2,3
4,5,6,7,8,1,2,3
Let's prove that it is indeed N steps. We know what element should be in the 1st place after the cut (the one at index M). We swap the 1st and Mth elements. Now we've got an array of length N-1 to arrange. Thus, we repeat the same steps N times and achieve the desired arrangement.

Now the code:

Java, Solution 1 - using LinkedList standard Java class:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.LinkedList;
import java.util.List;

/**
 * using the standard LinkedList from Collections
 */
public class CardDeckJ {
  private LinkedList<Integer> deck;
  private int n; // how many cards in the deck
```

```
  private int m; // how many cards to cut

  public CardDeckJ() {
    deck = new LinkedList<Integer>();
    n = 0;
  }

  /**
   * @throws Exception on a bad input
   */
  public void input() throws Exception  {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

    System.out.println("Enter numbers of cards in a deck:");
    String line = br.readLine().trim();
    n = Integer.parseInt(line);

    for(int i=0; i<n; i++) {
      System.out.printf("Enter next card (%d of %d):\n", i+1, n);
      line = br.readLine().trim();
      int card = Integer.parseInt(line);
      deck.add(card);
    }

    System.out.println("Enter how many cards to cut:");
    line = br.readLine().trim();
    m = Integer.parseInt(line);
    if(m<1 || m>=n)
      throw new Exception("invalid m");
  }

  public void cut() {
    List<Integer> start = deck.subList(0, m); // 1st part
    List<Integer> end   = deck.subList(m, n); // 2nd part
    end.addAll(start);                        // append
    for(int i=0; i<m; i++)                    // delete extras
      deck.removeFirst();
  }

  public void output() {
    System.out.print("deck: ");
    for(int card : deck)
      System.out.printf("%d, ", card);
    System.out.println();
  }

  public static void main(String[] args) throws Exception {
    CardDeckJ2 deck = new CardDeckJ2();
    deck.input();
    deck.cut();
    deck.output();
    System.out.println("end.");
  }
}
```

Java, Solution 2 - implementing linked list by hand, more efficient solution than one above:

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

/**
 * our own implementation of linked list (bare minimum)
 */
/* with generics
class Node<T> {
  T  v;         // value
  Node<T> next; // link to the next item

  Node(T val) {
    v = val;
    next = null;
  }

  Node<T> add(T val) {
    next = new Node<T>(val);
    return next;
  }

  Node<T> next() {
    return next;
  }

  T value() {
    return v;
  }
}
*/
// without generics for clarity
class Node {
  int  v;    // value
  Node next; // link to the next item

  Node(int val) {
    v = val;
    next = null;
  }

  Node add(int val) {
    next = new Node(val);
    return next;
  }

  Node next() {
    return next;
  }

  int value() {
    return v;
  }
}
```

```
public class CardDeckJ2 {
  private Node deck;
  private int n; // how many cards in the deck
  private int m; // how many cards to cut

  public CardDeckJ2() {
    deck = null;
    n = 0;
  }

  public void input() throws Exception  {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

    System.out.println("Enter numbers of cards in a deck:");
    String line = br.readLine().trim();
    n = Integer.parseInt(line);

    Node last = null;
    for(int i=0; i<n; i++) {
      System.out.printf("Enter next card (%d of %d):\n", i+1, n);
      line = br.readLine().trim();
      int card = Integer.parseInt(line);
      if(deck == null) {
        deck = new Node(card);
        last = deck;
      }
      else
        last = last.add(card);
    }

    System.out.println("Enter how many cards to cut:");
    line = br.readLine().trim();
    m = Integer.parseInt(line);
    if(m<1 || m>=n)
      throw new Exception("invalid m");
  }

  public void cut() {
    // "deck" points to the head of the list
    Node cutPoint = null;
    Node cutPointPrev = null; // just before the cut point
    Node end = null;
    Node card = deck; // current position
    for(int i=0; i<n; i++) {
      if(i == m-1)
        cutPointPrev = card;
      if(i == m)
        cutPoint = card;
      if(i == n-1) {
        end = card;
        assert(end.next == null);
      }
      card = card.next(); // advance
    }

    // now that we found key points, let's cut the deck
    end.next = deck;
```

```
    cutPointPrev.next = null;
    deck = cutPoint;
  }

  public void output() {
    System.out.print("deck: ");
    for(Node card=deck; card!=null; card=card.next())
      System.out.printf("%d, ", card.value());
    System.out.println();
  }

  public static void main(String[] args) throws Exception {
    CardDeckJ2 deck = new CardDeckJ2();
    deck.input();
    deck.cut();
    deck.output();
    System.out.println("end.");
  }

}
```

Python:

```
#!/usr/bin/python3

# cut the deck

import sys

class CardDeck:
  deck = [] # list of cards
  m = 0     # how many cards to cut

  # throws an exception on bad input
  def input(self):
    print("Enter numbers of cards in a deck:")
    n = int(sys.stdin.readline().strip())

    for i in range(n):
      print("Enter next card (%d of %d):\n" % (i+1, n));
      card = int(sys.stdin.readline().strip())
      self.deck.append(card)

    print("Enter how many cards to cut:")
    self.m = int(sys.stdin.readline().strip())
    if self.m<1 or self.m>=n:
      raise("invalid m")

  def cut(self):
    self.deck = self.deck[self.m : ] + self.deck[0 : self.m]

  def output(self):
    print(self.deck)

if __name__ == "__main__":
  deck = CardDeck()
  deck.input();
```

```
  deck.cut();
  deck.output();
  print("end.")
```

10 points:

For this month's assignment you will need to write a program to transform one string into another string, which consists of the same characters as the original string, but in a different order. You need to figure out what is the minimal number of operations necessary, if the **only** operation permitted is to take one of the characters in the string and move it to the front. For example, to transform ANT into TAN only one move in necessary: T to the front. To transform ABCD into ACBD two moves are needed: ABCD -> CABD -> ACBD. Your program should request an original and a target strings and then should print all the necessary transitions to transform the former into the latter. If transformation is not possible, your program should report so.

Solution:

First, the algorithm:

```
1. scan destination from tail to head and
   find corresponding chars in source in sequence
   "extra" chars will be remembered in "to_move"
2. arrange to_move in the insertion order
   (as they appear in destination)
3. move char in source from current place to front
```

Python:

```
#!/usr/bin/python3

import sys
from collections import defaultdict

# Note: we'll treat lists as arrays and use indices instead of pointers
#       for conciseness and clarity (at the expense of performance)
class RearrangeString:
  src = ""
  dst = ""

  def input(self):
    print("Enter the source string:")
    self.src = sys.stdin.readline().strip()
    print("Enter the destination string:")
    self.dst = sys.stdin.readline().strip()

  def transform(self):
    if len(self.src) != len(self.dst):
```

```
      return False

    # 1. scan destination from tail to head and
    # find corresponding chars in source in sequence
    # "extra" chars will be remembered in "to_move"
    to_move = defaultdict(list) # multimap: key=char => value=list_of_indices
    j = len(self.src)-1
    for i in range(len(self.dst)-1, -1, -1):
      ch = self.dst[i]
      while self.src[j]!=ch and j>=0:
        to_move[self.src[j]].append(j)
        j -= 1
      if j < 0: # we reached the start of the string
        break
      j -= 1

    # 2. arrange to_move in the insertion order
    # (as they appear in destination)
    move_queue = []
    for j in range(i, -1, -1):
      ch = self.dst[j]
      if ch in to_move:
        move_queue.append(ch)
        index = to_move[ch].pop(0)
        move_queue.append(index)
        if len(to_move[ch]) == 0:
          del to_move[ch]
      else:
        return False

    # 3. move char in source from current place to front
    print(self.src)
    j = 0
    while len(move_queue) > 0:
      ch    = move_queue.pop(0)
      index = move_queue.pop(0)
      self.src = ch + self.src[:index] + self.src[index+1:]
      print(self.src)
      j += 2

      # fix indices (if "ch" jumps over another the latter index increases)
      i = 0
      while i < len(move_queue):
        if move_queue[i+1] < index:
          move_queue[i+1] += 1
        i += 2

    return True

if __name__ == "__main__":
  solution = RearrangeString()
  solution.input()
  if solution.transform():
    print("transform is possible")
  else:
    print("transform is not possible")
  print("end.")
```