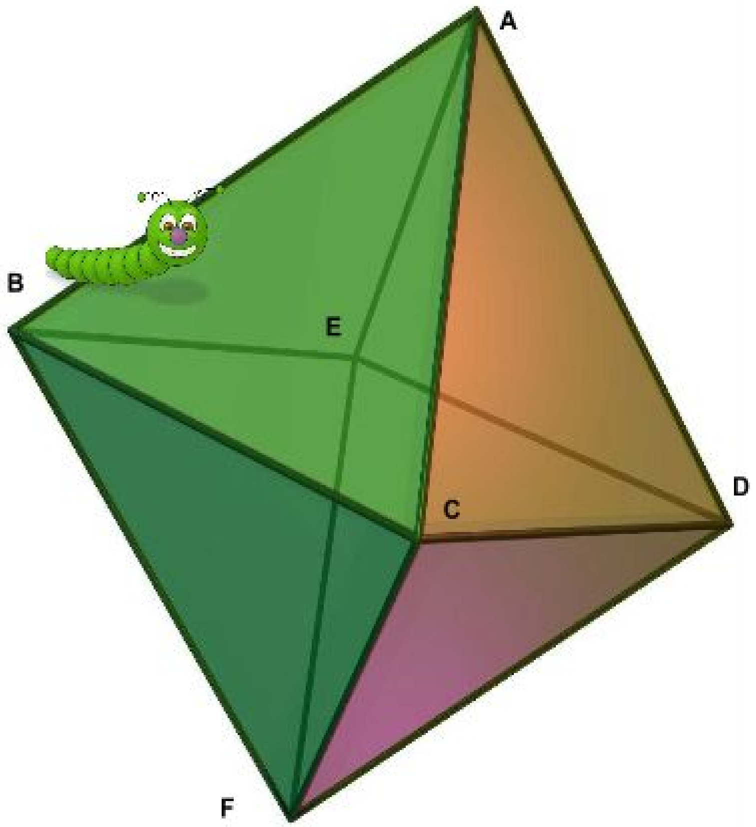
MATHEMATICS

5 points:

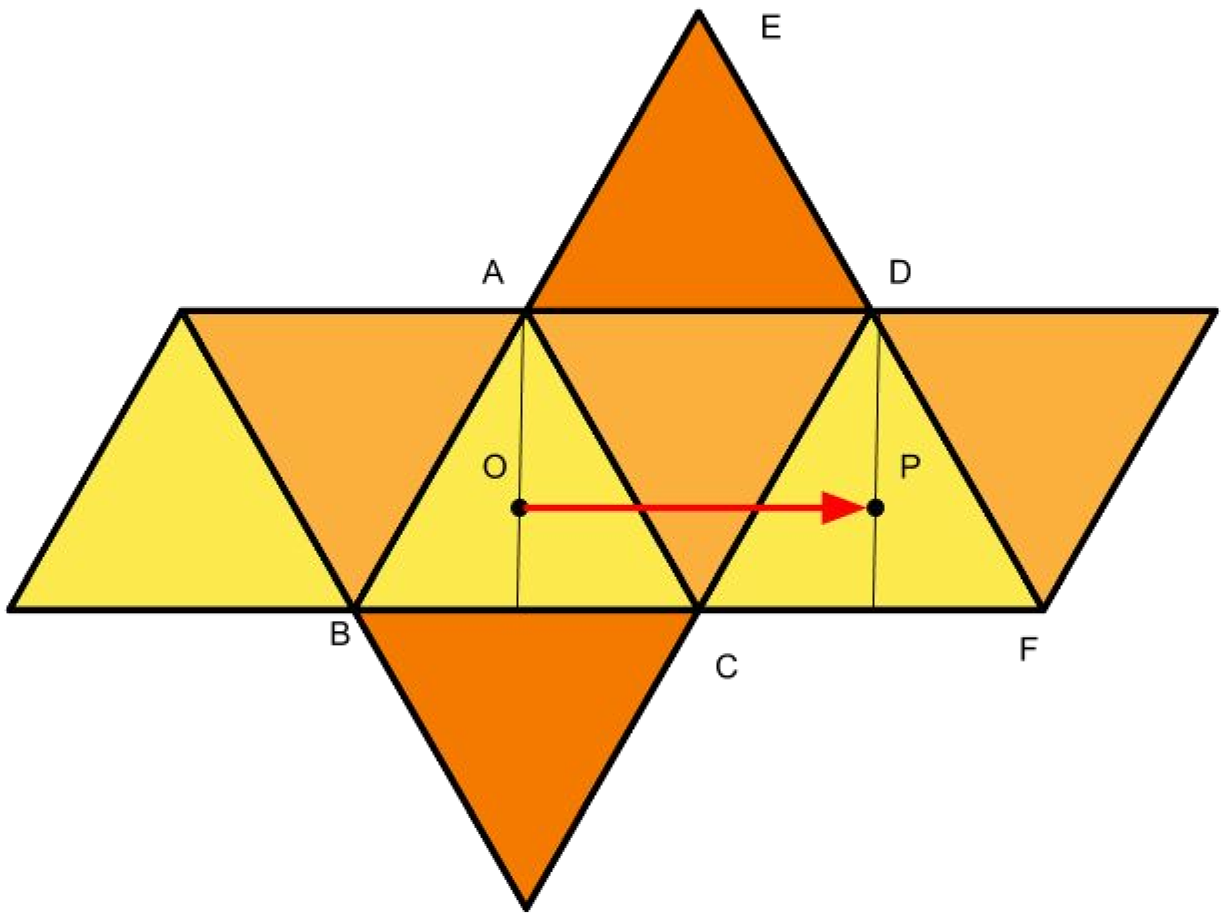
Each face of the octahedron ABCDEF in the Figure is an equilateral triangle with side ***a***. Caterpillar named Plato originally sits in the center of the face ABC, and wants to move to the center of face CDF. Plato can only move along the octahedron surface. What is the shortest distance that the Caterpillar has to travel to the destination?



Hint: Imagine that you want to fold the octahedron out of paper. In order to do this, you would have to cut out a special shape. Try to construct this shape, and identify on your picture the points between which the Plato has to travel.

Answer: a

Solution: Imagine that you want to fold the octahedron out of paper. In order to do this, you would have to cut out a shape made of triangles, as shown in the figure below:



Plato needs to travel from point O to point P, and it does not matter to it if the octahedron is already folded or still flat: the shortest route will be a line segment OP. Since OPAD is a rectangle, length of this route OP=a.

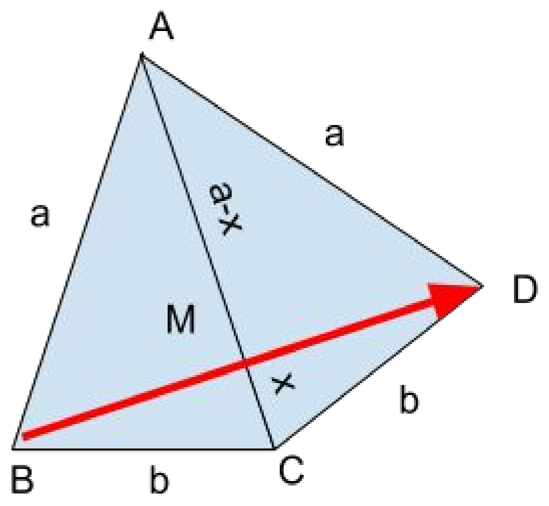
10 points:

Each face of the elongated octahedron ABCDEF in the Figure is an isosceles triangle with base ***b*** and sides ***a*** (for instance, **AB=AC=a** and **BC=b**). Caterpillar named Plato originally sits at the vertex B and wants to move to the opposite vertex D (BD is the diagonal of the square BCDE). Plato can only move on the octahedron surface. What is the shortest distance that the Caterpillar has to travel to the destination?

Hint: Imagine that you want to fold the octahedron out of paper. In order to do this, you would have to cut out a special shape. Try to construct this shape, and identify on your picture the points between which the Plato has to travel.

Answer: $2b\sqrt{1-\frac{b^2}{4a^2}}$

Solution: The idea of the solution is the same as for 5pt. problem, except the triangles are not equilateral but isosceles. Since Plato needs to move from point B to point D, only two triangles have to be considered: ABC and ACD. The segment BD is the shortest path, and it lies completely within these two triangles (however, there is another route of exact same length that goes through BAE and DAE triangles).



In the Figure, M is an intersection of segments BD and AC. We need to find BD, if AB=AC=AD=a, and BC=CD=b. There are two ways to solve this.

1. By symmetry, segments BD and AC are perpendicular. Therefore, in triangle ABC the altitude to side AC equals ½|BD|, and its area is $S = \frac{1}{4}|BD||AC| = \frac{1}{4}a|BD|$. This same area is, on the other hand, $S = \frac{1}{2}b\sqrt{a^2-(b/2)^2}$, where we have used the Pythagorean theorem to calculate the length of the altitude from the vertex A to the base $|BC| = b$. Equating these two expressions, we obtain $|BD| = 2\frac{b}{a}\sqrt{a^2-(b/2)^2} = 2b\sqrt{1-\frac{b^2}{4a^2}}$
2. Let MC=x. This means that MA=a-x. We can now use Pythagorean theorem for triangle BCM: $BM^2=b^2-x^2$;
   and for triangle BAM: $BM^2=a^2-(a-x)^2$;
   By comparing the two, we obtain one Equation:

$b^2-x^2=a^2-(a-x)^2$

It can be solved to find x:

$b^2-x^2=a^2-(a^2-2ax+x^2)$

$b^2=2ax$

$x=b^2/(2a)$

Now, we can find BM, and therefore the length of segment BD=2BM:

$$BM = \sqrt{b^2 - x^2} = \sqrt{b^2 - \tfrac{b^4}{4a^2}} = b\sqrt{1 - \tfrac{b^2}{4a^2}}$$

$$BD = 2b\sqrt{1 - \tfrac{b^2}{4a^2}}$$

PHYSICS

5 points:

Two wooden blocks of the same mass are glued together. The composite block is floating in water. What part of it is submerged into water if the densities of blocks are 500 kg/m3 and 1100 kg/m3, respectively?

Hint: Try to equate the Archimedean buoyancy force to the total weight of the composite block.

Answer: 11/16

Solution:

The total volume of the composite block is V=(m/500+m/1100). The mass of the water displaced by the submerged volume is M=2m=x 1000 V, where x is the fraction of the submerged volume we are looking for (1000 kg/m^3 is the density of the water). We have

2m = x 1000 (m/500+m/1100)

2=x 10 (1/5 + 1/11)

x = (2/10)/(1/5+1/11)=(1/10) 2 5 11/(5+11)=11/16

10 points:

A conical vessel has a small hole at the bottom (at the vertex of the cone). It is known that when V liters of water is placed into the vessel the velocity of fluid exiting the hole is U. What will be the velocity if one fills the vessel with $2V$ liters of water?

Hint: The exit velocity of the fluid from a small hole can be found in the following way. The hydrostatic pressure at depth h is equal to $P = \rho g h$, where ρ is the density of the fluid, and *h* is the height of the water level in the vessel. You can use Bernoulli Principle to relate this pressure *P* to the speed of flow v that it generates: $P = \rho v^2/2$.

Answer: $2^{1/6}U$

Solution:

The exit velocity of the fluid from a small hole can be found in the following way. The hydrostatic pressure at depth h is equal to $P = \rho g h$, where ρ is the density of the fluid, and *h* is the height of the water level in the vessel. You can use Bernoulli Principle to relate this pressure *P* to the speed of flow v that it generates: $P = \rho v^2/2$. By combining the two equations, we obtain $\rho v^2/2 = \rho g h$. Therefore, the speed of the flow is $v = \sqrt{2gh}$.

The volume of the cone is proportional to h^3 (height h times the area of the base which is in turn proportional to h^2). When volume is increased 2 times, the height is increased $2^{1/3}$ times. Hence, the velocity is increased by factor $\sqrt{2^{1/3}} = 2^{1/6}$.

CHEMISTRY

5 points:

Imagine you are playing the Escape the Room game. You found that to open a final lock you need to apply a weak electric signal to two wires connected to the doors. That will trigger the electronic device that controls the lock. The voltage of this signal should be 1 to 5 V. In the room, you found the following list of items:
1. A flashlight with a dead battery
2. A glass bowl with fruits (3 oranges, 2 lemons, 5 apples, grape, kiwi)
3 A rope
4 Scissors
5 A box with screws, zinc coated nails, stainless nuts and bolts
6 A glass of water
7 A wallet with 5 one dollar bills, and various coins (mostly dimes and pennies)
8 A bottle of liquid soap, a brush and toothpaste
9 A roll of copper wire
10 A bottle of Sprite

Which above listed items can you use to escape the room, and how concretely will you do that?

Hint: To escape the room, you need to make a battery. Try to google what the galvanic cell is, and which metals should be used to get needed voltage.

Solution: A simple electrochemical (galvanic) cell, a.k.a. “a battery” consists of two pieces of *different* metals separated by a liquid electrolyte (i.e. aqueous solution of some salt, or an acid, or a base). In this cell, one piece of metal becomes positive, whereas the other piece is negative. If you connect wires to these pieces of metal, you may detect a voltage between them. Which metals should you use to obtain the highest voltage? The voltage is highest when the difference in chemical activity of two metals is maximal. The information on the chemical activity of metals is summarized in so called “metal activity series” (you can easily google this series). The metals in the “potassium end” of the series are very active, and the metals in the “platinum side” are very unactive. As you can see, the most active metal available to you is zinc (the nails are zinc coated), and the most inactive metal is copper. Lemons, oranges, or other juicy fruits can serve as a good electrolyte.
Therefore, you need to do the following: using scissors, clean the surface of copper wire to make it shiny and stick it into a lemon or an orange (you also may use toothpaste to clean a penny, and to use a penny instead of wire), stick a nail into the same lemon about 1 cm apart from the copper wire (there is no need to clean a zinc coated nail). Now you can attach separate pieces of wire to both metals, and you got a simple electric

battery. Theoretically, the voltage of zinc-copper pair is about 1.1 V, but the actual voltage will be somewhat lower. To get more than 1 volt, you may connect two or more “lemon elements” in a series: a zinc coated nail from first lemon is connected to the copper wire in the second lemon, etc, and the external wires are connected to the copper wire (“electrode”) of the first lemon and the zinc nail of the last lemon. Accordingly, a two-lemon battery will produce up to 2 V (about 1.7), three-lemon battery produce about 2.5 V, etc.
By the way, strictly speaking it is the assembly of several individual cells that is called “a battery” (similar to an artillery battery, i.e. the group of cannons assembled together); a single galvanic cell is not a battery. Therefore, a standard AAA “battery” is not a battery (it contains just one galvanic cell), whereas 4.5 V battery or 9 V battery are real batteries (they contain 3 and 6 individual galvanic cells, accordingly, which are connected in series).

10 points:

A group of students, supervised by their teacher, Alice, and a technician, Bob, did some experiments in the school lab with the following solutions.
$LiNO_3$, $Pb(ClO_4)_2$, Na_2S, $Mg(ClO_4)_2$, $Ba(NO_3)_2$
After these experiments, Bob collected the remaining solutions into two waste bottles (some solutions went to one bottle, the bottle **A**, whereas other solution were poured to the another bottle, the bottle **B**. Since lead and barium are toxic heavy metals, he decided to put the “Hazardous waste! Heavy metals!” label on both bottles. However, Alice tell him to wait a little bit. “Look”, she said, “The solutions in both bottles are transparent and colorless. I think that means only one bottle contains heavy metals, whereas another bottle is a non-heavy metal waste. Let's check that”. She took a small amount of liquid from each bottle and mixed them in a test tube. A black precipitate formed immediately. “Ok, that is what I expected”, Alice said. “Let's make additional tests”. She poured a small amount of the waste solution **A** into a clean test tube and the same amount of the solution **B** into the another one. Then she took a bottle with some white crystalline powder from the shelf, dissolved it in water (to make approximately 5 percent solution), and added the solution she prepared to each test tube. A thick black precipitate formed in the first test tube, whereas the solution in the second test tube remained unchanged. “Aha”, Alice said. “One more test will clarify everything. Bob, I cannot find a bottle with 5% sodium sulfate solution. Can you please find it and bring it to me?” “Here it is” - Bob replied. “Thank you Bob”, Alice said. She added a small amount of sodium sulfate solution to each of two waste solutions. The solution **A** didn't change, but copious white precipitate formed in the solution **B**. “Look at that, Bob!” Alice said. “Definitely, only a bottle **B** contains heavy metal toxic waste, and the bottle **A** contains just ordinary low toxicity waste. That is a good news, because utilization of low

toxicity waste is much less complicated and costly. Please, label the bottles accordingly."
Explain please, why did Alice come to this conclusion, and which chemical ("a white powder") did she use? Please, draw the equations of the chemical reactions that took place in these tests.
Good luck!

Hint: To understand what concretely happened when Alice was mixing different solutions, one has to keep in mind that all chemicals Alice was dealing with were *salts*, i.e. ionic compounds that *dissociate* (produce ions) upon dissolution. That means there is no, e.g. $LiNO_3$ molecules in aqueous solution of these salts, there are Li^+ and NO_3^- ions instead. However, when the solutions containing the ions of certain types are mixed together, the ions may stick to each other to form insoluble ionic compounds that form a solid precipitate. The data on solubility of various ionic compounds are summarized in the solubility chart (google it).

Solution: All compounds Alice and Bob experimented with ($LiNO_3$, $Pb(ClO_4)_2$, Na_2S, $Mg(ClO_4)_2$, $Ba(NO_3)_2$) are salts, i.e. the ionic compounds that dissociate in water solution upon a cation and an anion. That means the mixture of all five salts contains three different anions (NO_3^-, ClO_4^-, and S^{2-}) and five different cations (Li^+, Na^+, Pb^{2+}, Mg^{2+}, Ba^{2+}). Two of these cations, barium and lead, are toxic heavy metal waste, and must be utilized properly.
To identify which bottle contained heavy metal ions, Alice asked herself: "Can these ions coexist in the aqueous solution?" The answer was: "Actually, no". Some of those ions cannot be present in solution in solution simultaneously with others. For example, lead ions, Pb^{2+}, and sulfide ions, S^{2-}, form a very insoluble solid, lead sulfide, which precipitates immediately when any soluble sulfide salt and soluble lead salt are mixed together. The detailed information on the ions that form insoluble precipitates with each other is collected in a table known as a "solubility chart". This table is easily googlable.
Being a good chemist, Alice knew the data on solubility of common salts, and she correctly concluded that the waste bottles cannot contain $Pb(ClO_4)_2$ and Na_2S simultaneously, because otherwise a solid precipitate would have formed. Alice mixed a small sample from each waste bottle, and the precipitate of lead sulfide has formed, thereby confirming her assertion: lead chlorate was in one bottle, and sodium sulfide was in another. The equation of this reaction is:

$Pb^{2+} + S^{2-} \rightarrow PbS$

In this equation, sodium and chlorate ions are omitted, because they do not participate in the reaction.
However, this test does not allow Alise to tell which of two bottles (**A** or **B**) contains toxic lead chlorate, and which bottle contains a relatively non-toxic sodium sulfide. From the solubility chart, Alice knows silver ions form a very insoluble solid when react with sulfide ions. She took silver nitrate ($AgNO_3$, a “white powder”), dissolved it in water, and added to both waste solutions. The black precipitate of silver sulfide formed only in one solution **A**, thereby confirming that sulfide ions are only in the bottle **A**, which meant lead was in the bottle **B**.
The next test is a test for barium ions. Alise knows barium ions form soluble salts with sulfide, nitrate, and chlorate ions, however, when mixed with sulfates, barium ions from a white precipitate of barium sulfate according to the equation:

$$Ba^{2+} + SO_4^{2+} \rightarrow BaSO_4$$

Alice took sodium sulfate (which is soluble in water, see the solubility chart) and added its solution to the waste solutions **A** and **B**. A precipitate formed only in the solution **B**, indicating that barium ions are only in the bottle **B**.
These observations can be summarized as follows:
- Lead ions and sulfide ions are in different waste bottles.
- Sulfide ions are in the bottle **A**, which means lead is in the bottle **B**.
- Barium ions are in the bottle **B**.

From these observations, Alice concluded only the bottle **B** contained the heavy metals (lead and barium), whereas the bottle **A** contained only low toxicity waste.

BIOLOGY

5 points:
It is known that animals with unusual coloration (albino or melanist) are sometimes born among the normal ones. As a rule, for many reasons, part of which are obvious, this unusual coloration poses a significant disadvantage negatively impacting the animal’s survival capabilities. Nevertheless, in some cases unusual coloration may be an advantage.
Please, tell what kind of advantages this abnormality may lead to. Describe as many negative consequences of the animal’s unusual coloration as possible.

Answer:

Disadvantages:

1. a breach of adaptive coloration (camouflage, mimicry, etc.);
2. animal "do not recognize" individuals of their own species;
3. albinism makes the tissue sensitive to ultraviolet;
4. albinism may increase the risk of cancer;
5. albinism can result in visual impairment;
6. melanism can lead to overheating;
7. discoloration can be a concomitant sign of a complex genetic disorder

Advantages:

1. unusual coloration may improve species adaptation & fitness (camouflage, mimicry, etc.);
2. white color reduces the possibility of overheating;
3. dark is better heated;
4. dark color offers better protection from UV radiation;
5. unusual color may have advantage for artificial selection.

10 points:

The immune system’s primary job is to protect the body from invading pathogens. That can be done because in every organism the outer surface of each cell has unique molecules (“antigens”) that serve as a signature of that organism, thereby allowing discrimination between the organism’s own cells and alien cells. The mechanism responsible for this discrimination (so called major immune-histocompatibility-complex, or MHC) combines a family of cells that are constantly crawling in between other cells and scanning them for the presence of antigens. The cells containing the organism’s own antigens are labeled as “self”, whereas the cell containing alien antigens are labelled as “non-self”. The cell labelled as “non-self” is attacked and destroyed.
Obviously, since MHS deals with a vast amount of different antigens, its malfunction is possible, when MHC erroneously recognises some of organism’s own cells as “non-self”, which leads to their destruction. That is a common mechanism of many autoimmune diseases, including diabetes, lupus, rheumatoid arthritis, etc. Interestingly, some autoimmune diseases, such as preeclampsia, develop in healthy women during pregnancy. As a rule, this disease ends after the woman delivers a child.

Given this information:

1. Why are women more vulnerable during pregnancy to autoimmune diseases?
2. If a woman bears two children (A, B), divorces, remarries, and then bears two more children (C, D), during which of her pregnancies is she most likely to develop preeclampsia? Why?

Answer:

1. The mother's genetics are distinct from that of the baby; the baby's proteins trigger an immune response that can attack not only the baby but also the mother, since the two systems are connected.
2. This occurs most frequently during the first pregnancy with a particular father (i.e., A and C). After the first pregnancy with a particular father, the woman's immune system adapts.

COMPUTER SCIENCE

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

5 points:

A Mad King wonders on a uni-dimensional chess board, with each move going Left (L) or Right (R). Given as an input an initial location of the King (as an integer 1 to 8), and a string of moves, which consists of letters L and R, write a program that determines whether the King returns to its original position at the end of the move sequence and print the result. Note that there is an abyss at both ends of the board, so that the move Left from position 1 and the move Right from position 8 lead to the Mad King falling off the board.

Examples:
Input: initial location: 2, moves: LRRL **Output**: yes
Input: initial location: 2, moves: LLRR **Output**: no

Answer:

Solution:
Java solution:
```
import java.util.Scanner;
public class WanderingKing{
      public static void main(String []args){
            Scanner input = new Scanner(System.in);
            int initialPosition = 0, done = 0, position;
            String moves;
            char move;
            while(done == 0) {
                  System.out.println("Enter initial position:");
                  initialPosition = input.nextInt();
                  if (initialPosition < 1 || initialPosition > 8) {
                        System.out.println("initialPosition should be between 1
and 8");
                  }
                  else {
                        System.out.println("Initial    position    entered    is
"+initialPosition);
                        done = 1;
                  }
            }

            done = 0;
            while(done == 0) {
                  System.out.println("Enter moves:");
                  moves = input.next().toUpperCase();
                  if (moves.isEmpty() == false) {
                  int len = moves.length();
                  position = initialPosition;
                  done = 1;
                  for (int i=0; i<len; i++) {
                        move = moves.charAt(i);
                        if (move == 'L') {
                              position--;
                              if (position < 1) {
                                    System.out.println("King   fell   off   the
board");
                                    break;
                              }
                        }
                        else if (move == 'R') {
                              position++;
                              if (position > 8) {
                                    System.out.println("King   fell   off   the
board");
                                    break;
                              }
                        }
                        else {
                              System.out.println("Bad   moves   entered.   Valid
moves are only L and R");
                              done = 0;
                              break;
                        }
                  }
                  if (done == 1) {
                        if (position == initialPosition)
                              System.out.println("King returned to its original
place");
                        else
                              System.out.println("King  did  not  return  to  its
original place");
                  }
            }
      }
  }
```

10 points:

Problem:
A Biosafety Level 4 undercover laboratory is located on a remote island in the Pacific Ocean. It is required by the NSA that a drone continuously patrols the perimeter of the island without ever flying directly over the island. You are to write a program that, given the map of the island, can generate a route for the drone to follow.

The map is provided to you on a square grid. You know that the island is grid-convex, meaning that any horizontal or vertical line on the map is either entirely over sea, or contains just one line segment that is over land. Additionally, one can get from any point on the island to any other point by moving horizontally and vertically only, and the drone can only fly horizontally and vertically also.

The map is provided to you as an array of N strings of length M, where a dot represents water and "#" represents land. Below is an example of a valid island, M = 6, N = 7, with the coordinate system defined

```
(0,0)   (5,0)
   ......
   ......
   ..##..
   .###..
   .#....
   .#....
   ......
(0,6)   (5,6)
```

Two examples of invalid islands (left one is not connected, right one is not convex) are below:

```
                   ......    ......
                   ....#.    ......
```

```
..##..    ..##..
.###..    .###..
.#....    .#....
.#....    .##...
......    ......
```

The drone accepts routing instructions in the form of a sequence of adjacent coordinates on the map. The drone can only fly horizontally and vertically, not diagonally. Your program should thus return a list of coordinates of the squares that the drone is to fly over. **As a reminder, the drone must follow the very perimeter of the island: It must fly over sea only, but every square it flies over must share either a side or a corner with land.**

For example, if the input is:

```
....
.##.
.#..
....
```

The output should be:
00 01 02 03 13 23 22 32 31 30 20 10

Answer:

Solution:

Python solution:

```
myMap = [   ".....",
            ".##..",
            "..#..",
            "....."]

Xdim = len(myMap[0])
Ydim = len(myMap)

#begin by finding a place to start:
for i in range(Ydim):
    Xinit = -1
    Yinit = i
    index = myMap[i].find("#")
    if index != -1: #-1 means "#" was not present in the string.
        Xinit = index - 1
        break #found coordinate, proceed
if Xinit == -1:
    Exception("no land on the myMap!")

#Now that we found the initial coordinate, proceed going around:

print "Initial position: ", Xinit, Yinit

#make first move, up. It is a valid move because we found the left top most
land and there is water all around the land:
Xcurr = Xinit
Ycurr = Yinit

lastMoveDir = "up"
Ycurr -= 1

#define some convenient functions:
def try_up():
    global Xcurr, Ycurr, lastMoveDir
    if Ycurr - 1 >= 0:
        if myMap[Ycurr - 1][Xcurr] == '.':
            Ycurr -= 1
            lastMoveDir = "up"
            return True
    return False

def try_down():
    global Xcurr, Ycurr, lastMoveDir
    if Ycurr + 1 < Ydim:
        if myMap[Ycurr + 1][Xcurr] == '.':
            Ycurr += 1
            lastMoveDir = "down"
            return True
    return False
```

```
def try_left():
    global Xcurr, Ycurr, lastMoveDir
    if Xcurr - 1 >= 0:
        if myMap[Ycurr][Xcurr - 1] == '.':
            Xcurr -= 1
            lastMoveDir = "left"
            return True
    return False

def try_right():
    global Xcurr, Ycurr, lastMoveDir
    if Xcurr + 1 < Xdim:
        if myMap[Ycurr][Xcurr + 1] == '.':
            Xcurr += 1
            lastMoveDir = "right"
            return True
    return False


#begin loop:
while (Xcurr != Xinit or Ycurr != Yinit):
    print Xcurr, Ycurr
    if lastMoveDir == "up":
        if try_right():
            pass
        elif try_up():
            pass
        elif try_left():
            pass
        else:
            print "error!!"

    elif lastMoveDir == "right":
        if try_down():
            pass
        elif try_right():
            pass
        elif try_up():
            pass
        else:
            print "error!!"

    elif lastMoveDir == "down":
        if try_left():
            pass
        elif try_down():
            pass
        elif try_right():
            pass
        else:
            print "error!!"

    elif lastMoveDir == "left":
        if try_up():
            pass
        elif try_left():
            pass
        elif try_down():
            pass
        else:
            print "error!!"
```

Java solution

```
The program consists of 4 parts:
1. input
2. path calculation
3. output

Algorithm:
1. find starting point
2. follow the perimeter counterclockwise keeping the land on the left
a. try cell to the left
b. if it's land then try straight
c. if it's land then try right
```

Convention:
The problem uses the coordinate system (x, y) with x going from left to right or from 0 to M, and y going from top to bottom or from 0 to N. We'll use more Java-esque notation [i,j] or [row,column] and we'll remap to (x,y) at output. Let's keep the map as an integer matrix map[i][j] where i is the row index, j is the column index, i goes from 0 to N excluding, j goes from 0 to M excluding, the value of 0 is for water, 1 is for land.

```
Pseudo code:
// find starting point
for each row i from 0 to N-1
      for each column j from 0 to M-1
            if map[i][j] = LAND then
                  // we found the left topmost land piece
                  p := i
                  q := j
                  stop iterating both loops
            endif
        endfor
endfor

// set starting point over water
q := q - 1

// let "route" be the drone path; it's an array of coordinates append to
route[] the first point (p, q)
do
      based on direction of drone (left, right, up or down)
      if cell to the left = WATER then
            adjust p or q
            turn direction 90 degree counterclockwise
      else if cell straight ahead = WATER then
            adjust p or q
      else
            // cell to the right must be water
            adjust p or q
            turn direction 90 degree clockwise
      endif

      next_route_point := (p, q)
while next_route_point != first_route_point
=================================================================
Point.java:
public class Point {
      private int p, q;

      public Point(int p, int q) {
            this.p = p;
            this.q = q;
      }
      public int getP() { return p; }
      public int getQ() { return q; }
      public void setP(int p) { this.p = p; }
      public void setQ(int q) { this.q = q; }

      public boolean equals(Object object) {
            if(object == null)
                  return false;
            if(!(object instanceof Point))
                  return false;
            Point other = (Point) object;
            if(other.getP()==this.p && other.getQ()==this.q)
                  return true;
            else
                  return false;
```

```
	}

	public int hashCode() {
		return 1000*p + q;
	}

	public String toString() {
		return "("+p+", "+q+")";
	}
}
=================================================================
DroneMain.java:

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.ArrayList;

public class DroneMain {
	public static final int WATER = 0;
	public static final int LAND = 1;
	public static final int DRONE = 2;
	public static final char WATER_MARK = '.';
	public static final char LAND_MARK = '#';
	public static final char DRONE_MARK = 'x';

	enum Direction { LEFT, DOWN, RIGHT, UP };

	private int map[][];
	/* private int map[][] =
		{ { 0, 0, 0, 0, 0, 0 },
		  { 0, 0, 0, 0, 0, 0 },
		  { 0, 0, 1, 1, 0, 0 },
		  { 0, 1, 1, 1, 0, 0 },
		  { 0, 1, 0, 0, 0, 0 },
		  { 0, 1, 0, 0, 0, 0 },
		   0, 0, 0, 0, 0, 0 } };
	*/

	private int p, q;
	private Direction direction;
	private ArrayList route;

	public DroneMain() {
		route = new ArrayList();
	}

	public static void drawMap(int[][] map) {
		for(int i=0; i<map.length; i++) {
```

```
            for(int j=0;  j<map[i].length; j++) {
                  if(j > 0)
                        System.out.print(' ');
            switch(map[i][j]) {
                  case WATER:
                        System.out.print(WATER_MARK);
                        break;
                  case LAND:
                        System.out.print(LAND_MARK);
                        break;
                  case DRONE:
                        System.out.print(DRONE_MARK);
                        break;
                  }
            }

            System.out.println();
      }
}

public boolean input() throws Exception {
      BufferedReader br = new BufferedReader(new InputStreamReader
(System.in));
      System.out.print("Enter number of rows: ");
      int N = Integer.parseInt(br.readLine());
      System.out.print("Enter number of columns: ");
      int M = Integer.parseInt(br.readLine());
      map = new int[N][M];

      System.out.println("Enter map (line by line, no spaces)");
      for(int i=0; i<N; i++) {
                  String str = br.readLine();
                  if(str == null)
                        break;
                  if(str.length() != M)
                        throw new Exception("number of columns != M");
                  for(int j=0; j<str.length(); j++) {
                        switch(str.charAt(j)) {
                        case WATER_MARK:
                              map[i][j] = WATER;
                              break;

                        case LAND_MARK:
                              map[i][j] = LAND;
                              break;

                        default:
                                throw   new   Exception("invalid   character:
      "+str.charAt(j));
             }
      }
}

System.out.println("Your input map is");
drawMap(map);
return true;

private void findStartingPoint() {
       for(int i=0; i<map.length; i++) {
             for(int j=0; j<map[i].length; j++) {
                   if(map[i][j] == LAND) {
                         p = i;
                         q = j;
                         q--; // set starting point over water
                         return;
                   }
             }
       }
 }

      public void calcPath() {
            findStartingPoint();
            route.add(new Point(p,q));
            direction = Direction.DOWN;

            int count;
            int maxCount = 1000;
            for(count=0; count<maxCount; count++) {
                  switch(direction) {
                  case DOWN:
                   if(map[p][q+1] == WATER) {
                         q++;
                         direction = Direction.RIGHT;
                   }
                         else if(map[p+1][q] == WATER) {
                         p++;
                   }
                   else {
                         assert(map[p][q-1] == WATER);
                         q--;
                         direction = Direction.LEFT;
                   }
                   break;

                  case RIGHT:
```

```
 if(map[p-1][q] == WATER) {
      p--;
      direction = Direction.UP;
 }
 else if(map[p][q+1] == WATER) {
      q++;
 }
 else {
      assert(map[p+1][q] == WATER);
      p++;
      direction = Direction.DOWN;
 }
 break;

case UP:
 if(map[p][q-1] == WATER) {
      q--;
      direction = Direction.LEFT;
 }
 else if(map[p-1][q] == WATER) {
      p--;
 }
 else {
      assert(map[p][q+1] == WATER);
      q++;
      direction = Direction.RIGHT;
 }
 break;

case LEFT:
 if(map[p+1][q] == WATER) {
      p++;
      direction = Direction.DOWN;
 }
 else if(map[p][q-1] == WATER) {
      q--;
 }
 else {
      assert(map[p-1][q] == WATER);
      p--;
      direction = Direction.UP;
 }
 break;
}

Point nextRoutePoint = new Point(p, q);
if(nextRoutePoint.equals(route.get(0)))
      break;
```

```
                else
                        route.add(nextRoutePoint);
            }
            if(count >= maxCount)
                System.out.println("exceeded maxCount: either you have a bug
            and there's possibility of infinite loop or your island is big and
            you should increase maxCount");
        }

        public void outputPath() {
            System.out.println("The calculated drone route is");
            for(Point point : route) {
                // reverse coordinates
                int x = point.getQ();
                int y = point.getP();
                System.out.printf("%d%d ", x, y);
            }
            System.out.println();

            int map2[][] = map;
            for(Point point : route) {
                int p2 = point.getP();
                int q2 = point.getQ();
                map2[p2][q2] = DRONE;
            }
            drawMap(map2);
        }

        public static void main(String[] args) throws Exception {
            DroneMain drone = new DroneMain();
            if(!drone.input()) throw new Exception("invalid input");
            drone.calcPath();
            drone.outputPath();
        }
}
```