TASK 4.2.1: "HAMILTON'S ROBOT"
==============================

On a plane there are given N positions P1, P2, ..., PN with integer coordinates (X1,Y1), (X2,Y2), ..., (XN,YN).

A robot should move through all these positions starting at P1. It should come to each position only once with the exception of P1 which also has to be the position at the end of the tour.

There are constraints on the robot's movements. It can only move along straight lines. From P1 it can start in any direction. Reaching one of the Pi, before moving on to another position it must turn 90 degrees either to the left or to the right.

A robot program consists of five types of statements:

1. "ORIENTATION Xk Yk": usable as the first statement only. The robot turns to the direction of the position Pk (k between 2 and N).
2. "MOVE-TO Xj Yj"   : if the robot can reach Pj without changing its current orientation, then it moves to the position Pj (j between 1 and N). Otherwise the statement is not executable.
3. "TURN-LEFT"   : the robot changes its orientation 90 degrees to the left.
4. "TURN-RIGHT"   : the robot changes its orientation 90 degrees to the right.
5. "STOP"   : deactivates the robot. This is the necessary last statement of each robot program.

PROBLEM STATEMENT
=================

Implement a program that does the following:

1. Read the value of N and the coordinates for N given positions from an ASCII input file (see Example) and display the data on the screen.
2. Develop a robot program for a valid tour through all positions (as defined above) if one exists.
3. If there is no possible tour, the robot program must consist just of the "STOP"-statement.
4. Display on the screen, whether a tour is possible or not and, if there exists one, its length (rounded, 2 digits after the decimal point). The length of a tour the sum of the lengths of the straight line pieces.
5. Write the robot program to an ASCII output file exactly as is shown in Example.

TECHNICAL CONSTRAINTS
=====================

Constraint-1: Put your solution program into an ASCII text file named

"C:\IOI\DAY-2\421-PROG.xxx". Extension .xxx is:

```
- .BAS for BASIC programs, .C   for C      programs,
- .LCN for LOGO  programs, .PAS for PASCAL programs.
```

Constraint-2: The name of the ASCII input file for reading the positions from must be "C:\IOI\DAY-2\421-ROBO.IN".

Constraint-3: The name of the ASCII output file for writing the robot program to must be "C:\IOI\DAY-2\421-ROBO.OU".

Constraint-4: Program must reject inputs where N is less than 4 or greater than 16, without trying to find a tour!

EXAMPLE(S)
==========

Input: An input file contains in the first line the value for N and in the following N lines the X and Y coordinates of the selected positions, for example:

```
4
2 -2
0 2
-1 -1
3 1
```

Output: For these 4 positions one shortest robot program with length = 12.65 is:

```
ORIENTATION 3 1
MOVE-TO 3 1
TURN-LEFT
MOVE-TO 0 2
TURN-LEFT
MOVE-TO -1 -1
TURN-LEFT
MOVE-TO 2 -2
STOP
```

SAMPLE FILES
============

We provide these correct files with the above input and output for your convenience:
"C:\IOI\DAY-2\421-ROBO.IN" and "C:\IOI\DAY-2\421-ROBO.OU".

WARNING: Successful execution of your program with this example does not necessarily guarantee that your program is correct !!!

CREDITS
=======

```
Read input data correctly from every file and display it....  5 points
Algorithm for computing a valid tour ok ................... 30 points
Generated robot program syntactically correct,
   if tour does not exist ................................. 10 points
Generated robot program syntactically correct,
   if tour does exist ..................................... 15 points
Screen display gives all required information .............  5 points
Displayed length of computed tour correct ................. 10 points
Robot program correctly written to a file ................. 10 points
```

```
Technical constraints obeyed .............................. 15 points
----------------------------------------------------------------------
                                                    maximal 100 points
```

Problem Chosen for the second session (5 hours)

***TASK 4.2.2: "CLIMBING A MOUNTAIN"

A mountain climbers club has P members, numbered from 1 to P. Every member climbs at the same speed and there is no difference in speed between climbing up and down. Climber number i consumes C(i) units of SUPPLIES per day and can carry at most S(i) such units. All C(i) and S(i) are integer numbers.

Assume that a climber with a sufficient amount of supplies would need N days to reach the top of the mountain. The mountain may be too high, so that a single climber cannot carry all the necessary supplies. Therefore a GROUP of climbers starts at the same place and at the same time. A climber who descends prematurely before reaching the top gives his unneeded supplies to other climbers. The climbers do not rest during the expedition.

The PROBLEM is to plan a schedule for the climbing club. At least one climber must reach the top of the mountain and all climbers of the selected group return to the starting point.

PROBLEM STATEMENT

Implement a program which does the following:

1. Read from the keyboard the integer number N of days needed to arrive at the top, the number P of climbers in the club, and (for all i from 1 to P) the numbers S(i) and C(i).
   You may assume that the inputs are integers.
   Reject inputs that make no sense.

2. Try to find a schedule for climbing the mountain. Determine a possible group a(1), ..., a(k) of climbers who should participate in the party and (for all j from 1 to k) the number M(j) of supplies which climber a(j) carries at the start.
   Note that there may not exist a schedule for all combinations of N and the S(i) and C(i).

3. Output the following information on the screen:
   a) the number k of climbers actually participating in the party,
   b) the total amount of supplies needed,
   c) the climber numbers a(1), .., a(k),
   d) for all a(j), j between 1 and k, the
      initial amount M(j) of supplies to carry for climber a(j),
   e) the day D(j) when climber a(j) starts descending.

4. A schedule is OPTIMAL if
   a) the number of participating climbers is minimal and
   b) among all groups satisfying condition a) the total of consumed supplies is minimal.
   Try to find a nearly optimal schedule.

TECHNICAL CONSTRAINTS

Constraint-1: Put your solution program into an ASCII text file named "C:\IOI\DAY-2\422-PROG.xxx". Extension .xxx is:

- .BAS for BASIC programs, .C   for C   programs,
- .LCN for LOGO   programs, .PAS for PASCAL programs.

Constraint-2: Programs must reject inputs where N is less than 1 or greater than 100. P must be not less than 1 and not greater than 20.

EXAMPLE(S)

The following could be a dialogue with your program:

```
   Days to arrive to top:  4
   Number of club members: 5
   Maximal supply for climber 1 : 7
   Daily consumption for climber 1 : 1
   Maximal supply for climber 2 : 8
   Daily consumption for climber 2 : 2
   Maximal supply for climber 3 : 12
   Daily consumption for climber 3 : 2
   Maximal supply for climber 4 : 15
   Daily consumption for climber 4 : 3
   Maximal supply for climber 5 : 7
   Daily consumption for climber 5 : 1

   2 climbers needed, total amount of supplies is 10.
   Climber(s) 1, 5 will go.
   Climber 1 carries 7 and descends after 4 day(s)
   Climber 5 carries 3 and descends after 1 day(s)

   Plan another party (Y/N) Y

   Days to arrive to top:  2
   Number of club members: 1
   Maximal supply for climber 1 : 3
   Daily consumption for climber 1 : 1
   Climbing party impossible.
   Plan another party (Y/N) N

   Good bye
```

SAMPLE FILES

For your convenience, some files containing test data and correct sample output have been prepared; please look into the directory "C:\IOI\DAY-2".

WARNING: Successful execution of your program with these examples does not necessarily guarantee that your program is correct !!!

CREDITS

User dialogue as illustrated above.........................	10 points
Find a solution for the special case where all C(i)=1 and all S(i) are equal	20 points
Find a solution for general case	20 points
Find a nearly optimal solution for general case	30 points
Detect unsolvable situations	10 points
Technical constraints obeyed	10 points

maximal 100 points

TASK 4.2.3: "RUBIK'S TOOLKIT"

This problem is based on the puzzle game "Rubik's cube".

If you already know Rubik's cube you may skip this paragraph and the next one. Rubik's cube is a cube that consists of 3 x 3 x 3 smaller cubes. Initially each of the six faces of Rubik's cube is coloured uniformly in a different colour; we call this the initial cube. Every face of Rubik's cube consists of 3 x 3 faces of a layer of nine smaller cubes.

Imagine you are looking at any of the six faces of Rubik's cube. The layer of 3 x 3 smaller cubes you see can be rotated by a multiple of 90 degrees, where the axis of rotation is orthogonal to the face and goes through its centre. The result is another 3 x 3 x 3 cube where the colour pattern of the face you are looking at has been rotated and the colour patterns of the four neighbouring faces have changed.

In our problem the faces of the cube are given names instead of colours: U=Up, R=Right, F=Front, B=Back, L=Left and D=Down. Any move sequence to turn the cube may be described as a string of the letters {U, R, F, B, L, D} where each letter stands for a basic rotation: the 90 degrees clockwise rotation of the corresponding face.

PROBLEM STATEMENT with EXAMPLE(S)

Write a program that allows the user to repeatedly solve any of the given three subproblems in any order. You may assume that the length of each input string is at most 35.

1. This subproblem is the translation of a given move sequence into a move sequence where no primitive rotation is applied more than 3 times in sequence. Your algorithm should reject non-legal input sequences. Some examples are provided for clearness:

Input		Output
L	-->	L
LL	-->	LL
LLL	-->	LLL
LLLL	-->	"the empty sequence"
LLLLL	-->	L
LLRRRFFFFRLB	-->	LLLB
HELLO	-->	"error"

2. The second subproblem is to find out whether two given move sequences yield the same result when applied to the initial cube. The examples may illustrate this:

Input,	Input,	Output
1st sequence	2nd sequence	
RL	LR	yes
RU	UR	no
RRFFRRFFRRFFRRFF	FFRRFFRR	yes

RRFFRRFFRRFFRRFF	RRFFRRFF	no

3. The third subproblem is to determine how many times a given move sequence has to be applied to the initial cube until the cube is in its initial state again. The smallest such number greater zero is sought.

   We provide some examples:

Input	Output
L	4
DD	2
BLUB	36
RUF	80
BLUFF	180

TECHNICAL CONSTRAINTS

Constraint-1: Put your solution program into an ASCII text file named
"C:\IOI\DAY-2\423-PROG.xxx". Extension .xxx is:
- .BAS for BASIC programs, .C for C programs,
- .LCN for LOGO programs, .PAS for PASCAL programs.

SAMPLE FILES

none

CREDITS

Main menu and user dialogue o.k.	15 points
Subproblem 1: Transformation o.k.	20 points
Rejects wrong inputs	10 points
Subproblem 2: Correctness	25 points
Subproblem 3: Correctness	25 points
Technical constraints obeyed	5 points
	maximal 100 points