The Short and the Tall

R x C students sit in an array of R rows by C columns. The shortest in each row stands up and the tallest of these, A, remains standing. The tallest student in each column stands up and the shortest of these, B, remains standing.

If $A \neq B$, is $A > B$?

If $A = B$, we have a Saddle-Point in the array

e.g. Matrix of Integers

				Min in Row	
	5	8	9	5	-- A
	6	7	2	2	
	7	1	9	1	
Max in Col	7	8	9		
	B				

In this case, not ($A > B$)

Saddle-Point

Given a RxC Matrix, A, i.e. R rows and C columns we define a Saddle-Point as

Saddle_Pt (A(i,j))

≡ A(i,j) is the minimum of Row i and the maximum of Col j.

e.g.

	1	2	3	
	4	5	6	-- 7 is Saddle_Pt.
	7	8	9	-- at position (3,1)

There may be more than one Saddle-Pt,

e.g. an all zero matrix.

Theorem 1.

if A(i,j) and A(s,t) are Saddle-Points then A(i,j) = A(s,t)

Proof:

A(i,j) = Min of row I
tf. (All k | 1 ≤ k ≤ C : A(i,j) ≤A(i,k))

In particular,
A(i,j) ≤A(i,t) -- A(i,j) is Min of row I
but A(i,t) ≤A(s,t) -- A(s,t) is Max of col t
tf. A(i,j) ≤A(s,t)

Similarly, A(s,t) ≤ A(i,j)
tf. A(i,j) = A(s,t)

End Proof.

Problems

1. Find the positions of all Saddle Points.
2. Find just one Saddle Point and its position

Let

MinRow(i) = Minimum of Row I
MaxCol(j) = Maximum of Col j

tf.

Saddle_Pt(A(i,j)) ≡ A(i,j) = MinRow(i) ∧ A(i,j) = MaxCol(j)
i.e. Saddle_Pt(A(i,j)) ≡ MinRow(i) = MaxCol(j)

To find all Saddle-Points in A, find all (i,j) s.t. MinRow(i) = MaxCol(j)

To find just one, we could start by finding all and exit having found the first.

But, we consider an alternative solution which will justify the claim in the 'The Short and the Tall" that if A=B then we have a Saddle-Point.

Notation:

Let f be a function
M = (Max k | 1 ≤ k ≤ n : f(k))
≡ (Exists k | 1 ≤ k ≤ n & M=f(k)) ∧ (All k | 1 ≤ k ≤ n : f(k)≤M)

Similarly for (Min k | 1 ≤ k ≤ n : f(k))

Theorem 2.

If Saddle_Pt(A(i,j)) then A(i,j) = (Max k | 1≤k≤n : MinRow(k))
also
If Saddle_Pt(A(i,j)) then A(i,j) = (Min k | 1≤k≤n : MaxCol(k))

Proof:

A(i,j) = MinRow(i) as Saddle_Pt(A(i,j))
also A(i,j) = MaxCol(j) as Saddle_Pt(A(i,j))

Show for all i≤k≤ R, MinRow(k) ≤ A(i,j)

MinRow(k) ≤ A(k,j) -- Min of row k,
≤ A(i,j) -- Max of col j

tf. A(i,j) = (Max k | 1≤k≤R : MinRow(k))

Similarly,
A(i,j) = (Min k | 1≤k≤C : MaxCol(k))

End Proof

Theorem 3.

Let MinRow(mx) = (Max k | 1≤k≤R: MinRow(k)) -- A in "Short & Tall"
Let MaxCol(mn) = (Min k | 1≤k≤C : MaxCol(k)) -- B in "Short & Tall"

If MinRow(mx) = MaxCol(mn) then Saddle_Pt(A(mx,mn))

Proof:

Assume MinRow(mx) = MaxCol(mn),
Show Saddle_Pt(A(mx,mn)).

MinRow(mx) = A(mx,j) some j: 1≤ j≤C
also MaxCol(mn) = A(i,mn) some i: 1≤i≤R

Consider A(mx,mn)
A(mx,j) ≤A(mx,mn) -- A(mx,j) = MinRow(mx)
Also A(mx.mn) ≤A(i,mn) -- A(i,mn) = MaxCol(mn)

tf. A(mx,j) ≤ A(mx,mn) ≤ A(i,mn)

From assumption,
MinRow(mx) = MaxCol(mn)
i.e. A(mx,j) = A(i,mn)
tf. A(mx,mn) = MinRow(mx)

also A(mx,mn) = MaxCol(mn)
tf. Saddle_Pt(A(mx,mn))

End Proof

In Eiffel,

```
class  SADDLE

creation
        make

feature

        mat: MATRIX [INTEGER];

        make is
                local
                        r, c: INTEGER
                do
                        io.put_string ("%N Enter absolute/full filename: ");
                        io.read_word;
                        file2matrix (io.last_string);
                        print_matrix (mat, mat.rows, mat.cols);
                        io.put_string ("Looking for one Saddle Point .. %N");
                        one_saddle (mat);
                        io.put_new_line;
                        io.put_string ("Looking for All Saddle Points ....%N");
                        all_saddle (mat)
                end ;

        one_saddle (m: MATRIX [INTEGER]) is
                        -- Max of Cols of M
                local
                        mni, mxi: INTEGER;
                        mnr, mxc: VECTOR [INTEGER];
                        tr: MATRIX [INTEGER]
                do
                        mnr := m.min_row;
                        tr := m.transpose;
                        mxc := tr.max_row;
                        mxi := mnr.max_index;
                        mni := mxc.min_index;
                        if  mnr.item (mxi) = mxc.item (mni) then
                                print_saddle (m, mxi, mni)
                        end
                end ;
```

```
all_saddle (m: MATRIX [INTEGER]) is
        -- Max of Cols of M
    local
        mnr, mxc: VECTOR [INTEGER];
        tr: MATRIX [INTEGER];
        i, j: INTEGER
    do
        mnr := m.min_row;
        tr := m.transpose;
        mxc := tr.max_row;
        from
            i := 1
        until
            i > m.rows
        loop
            from
                j := 1
            until
                j > m.cols
            loop
                if  mnr.item (i) = mxc.item (j) then
                    print_saddle (m, i, j)
                end ;
                j := j + 1
            end ;
            i := i + 1
        end
    end ;

print_saddle (m: MATRIX [INTEGER]; i, j: INTEGER) is
    do
        io.put_integer (m.item (i, j));
        io.put_string (" is a Saddle_Point at ");
        io.put_integer (i);
        io.put_character (' ');
        io.put_integer (j);
        io.put_new_line
    end ;
```

```
file2matrix (fname: STRING) is
        -- Input from file, fname, into the matrix, mat.
        -- First 2 numbers give #rows and #cols
    local
        in_file: PLAIN_TEXT_FILE;
        i, j, r, c: INTEGER;
        x: INTEGER
    do
        !! in_file.make_open_read (fname);
        in_file.read_integer;
        r := in_file.last_integer;
        in_file.read_integer;
        c := in_file.last_integer;
        !! mat.make (r, c);
        from
            i := 1
        until
            i > r
        loop
            from
                j := 1
            until
                j > c
            loop
                in_file.read_integer;
                x := in_file.last_integer;
                mat.put (x, i, j);
                j := j + 1
            end ;
            i := i + 1
        end ;
        in_file.close
    end ;
```

```
        print_matrix (m: MATRIX [INTEGER]; r, c: INTEGER) is
                -- M has r rows and c columns
                -- i.e. M is of height r and width c
            local
                i, j: INTEGER
            do
                from
                    i := 1
                until
                    i > r
                loop
                    from
                        j := 1
                    until
                        j > c
                    loop
                        io.put_integer (m.item (i, j));
                        io.put_character (' ');
                        j := j + 1
                    end ;
                    io.put_new_line;
                    i := i + 1
                end ;
                io.put_new_line
            end ;

end  -- class SADDLE
```