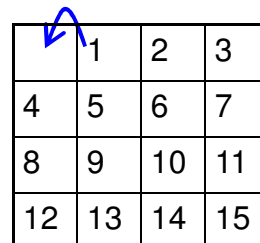


CMSC 330: Organization of Programming Languages

Project 3 – Sliding Puzzle

Sliding Puzzle

- ▶ Numbered tiles in a board
 - One empty space
- ▶ Move
 - Slide adjacent tile into space
- ▶ Continue until
 - All tiles are in order
 - Space in top left corner



	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15



Puzzle Representation

- ▶ Represent puzzle as int list
- ▶ 2D array (in row-major order) • (0,0) (0,1) (0,2)
(1,0) (1,1) (1,2)
(2,0) (2,1) (2,2)
- ▶ 2D coordinates stored in 1D • (0,0) (0,1) (0,2) (1,0) (1,1) (1,2) (2,0) (2,1) (2,2)
- ▶ 1D positions • 0 1 2 3 4 5 6 7 8
- ▶ 2D (x,y) coordinate $\leftrightarrow$ 1D position
 - X goes down, Y goes right (opposite normal behavior)

CMSC 330

3

Puzzle Representation in OCaml

- ▶ int list = board
 - [0;1;2;3] (0 = space, sorted = solved)
- ▶ int list list = list of boards
 - [[1;0;2;3]; [0;1;2;3]]
 - May indicate a **solution** if
 - Adjacent boards result of a single move
 - Boards start with original configuration & end at solved board
- ▶ int list list list = list of solutions
 - [[[1;0;2;3]; [0;1;2;3]] ; [[2;1;0;3]; [0;1;2;3]]]

CMSC 330

4

Project 3

- ▶ Implement utility functions in OCaml
 - Together can be used to solve puzzle
- ▶ Learn to use
 - Lists
 - Recursion
- ▶ OCaml modules
 - May use functions in Pervasives
 - May not use any other modules (e.g., List, Array, Set)

CMS 330

5

Project 3

- ▶ Functions use currying
 - `index x v` `'a list -> 'a -> int`
- ▶ In general, assume legal inputs
 - Up to you to ensure used with legal inputs in solver

CMS 330

6

Project Files

- ▶ Your code
 - puzzle.ml
- ▶ Public test cases
 - testRecursion1.ml, testRecursion2.ml
 - testHigherOrder.ml
 - testPuzzle1.ml, testPuzzle2.ml, testPuzzle3.ml
 - testSolve1.ml, testSolve2.ml
- ▶ Utility files
 - testUtils.ml - OCaml functions used in test cases
 - goTest.rb - Ruby script to run test cases

CMS3 330

7

Using OCaml

- ▶ Run interpreter in shell
 - Go to directory p3 (from p3.zip) containing project files
 - E.g., `cd c:\Users\myname\Desktop\p3`
- ▶ Type `ocaml testRecursion1.ml`
 - Runs test code in OCaml interpreter
 - Will include puzzle.ml in testRecursion1.ml
 - Should produce same output as in testRecursion1.out
 - Typing `ocaml puzzle.ml` won't produce any output
- ▶ Type `ruby goTest.rb`
 - Runs each public test case OCaml interpreter
 - First uncomment line selecting `fc` / `diff` for comparisons

CMS3 330

8