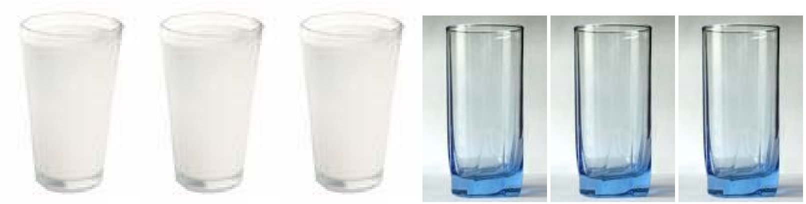
# A Peek at Programming (http://www.comp.nus.edu.sg/~tantc/bingo)

*or, problem solving in Computer Science*

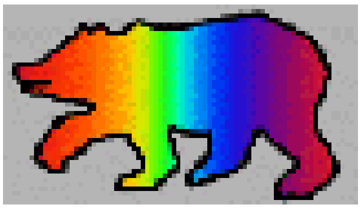
Aaron Tan
School of Computing (http://www.comp.nus.edu.sg)
National University of Singapore (http://www.nus.edu.sg)

June 2010

## Warm-up #1: Glasses of milk



## Warm-up #2: Bear

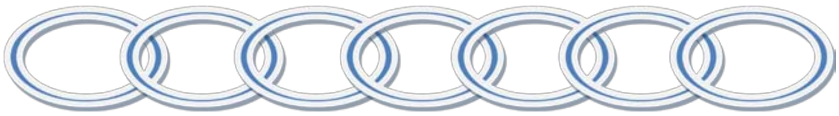


## Warm-up #3: Mad scientist

In how many ways can a chain of length 6 be constructed?

In how many ways can a chain of length *n* be constructed?

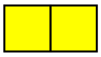
## Warm-up #4: Silver chain

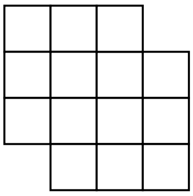


## Warm-up #5: Dominoes

Show that it is not possible to cover a 4×4 board (with 2 opposite corners removed) with dominoes.

Show that it is not possible to cover an *n*×*n* board (with 2 opposite corners) removed with dominoes.

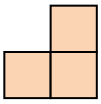


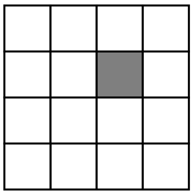


## Warm-up #6: Triominoes

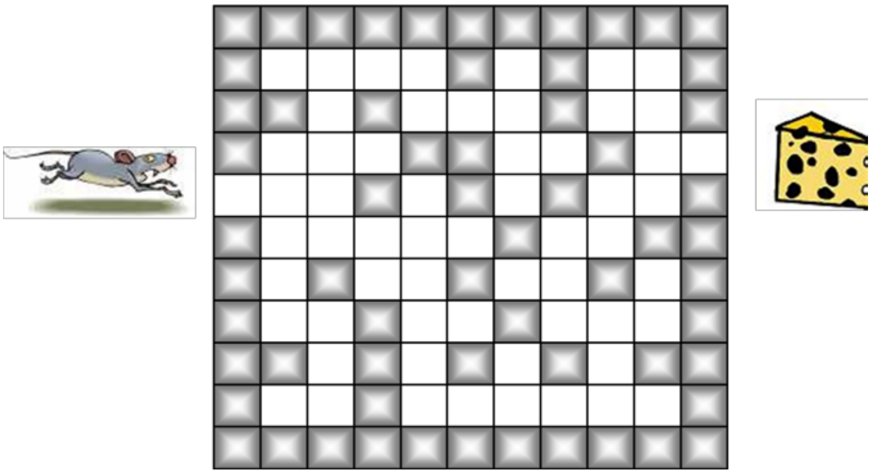
Show that a 4×4 board with a hole can be covered with triominoes.

Show that any $2^n \times 2^n$ board with a hole can be covered with triominoes.





## Algorithmic Problem Solving #1: Maze



## Algorithmic Problem Solving #2: Sudoku

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | | | 7 | | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | 3 | | | | | 4 | 9 | |
| 8 | | | | | 5 | 1 | 6 | |
| | 6 | | | | 9 | | | |
| | | | 6 | | 8 | | 1 | 7 |
| | | 5 | | | | 9 | | |
| 6 | 8 | | 9 | | 7 | | | |
| | | | 5 | | | | 4 | |
| | 5 | 7 | 2 | | | | | 9 |
| | 4 | 8 | | | | | 5 | 2 |

## Algorithmic Problem Solving #3: MasterMind

Colours: (R)ed, (B)lue, (G)reen, (Y)ellow, (C)yan, (M)agenta

|  | ○ | ○ | ○ | ○ | Sinks | Hits |
|---|---|---|---|---|---|---|
| 10. | ○ | ○ | ○ | ○ | □ | □ |
| 9. | ○ | ○ | ○ | ○ | □ | □ |
| 8. | ○ | ○ | ○ | ○ | □ | □ |
| 7. | ○ | ○ | ○ | ○ | □ | □ |
| 6. | ○ | ○ | ○ | ○ | □ | □ |
| 5. | ○ | ○ | ○ | ○ | □ | □ |
| 4. | ○ | ○ | ○ | ○ | □ | □ |
| 3. | ○ | ○ | ○ | ○ | □ | □ |
| 2. | ○ | ○ | ○ | ○ | □ | □ |
| 1. | ○ | ○ | ○ | ○ | □ | □ |

|  | ○ | ○ | ○ | ○ | Sinks | Hits |
|---|---|---|---|---|---|---|
| 10. | ○ | ○ | ○ | ○ | □ | □ |
| 9. | ○ | ○ | ○ | ○ | □ | □ |
| 8. | ○ | ○ | ○ | ○ | □ | □ |
| 7. | ○ | ○ | ○ | ○ | □ | □ |
| 6. | ○ | ○ | ○ | ○ | □ | □ |
| 5. | ○ | ○ | ○ | ○ | □ | □ |
| 4. | ○ | ○ | ○ | ○ | □ | □ |
| 3. | ○ | ○ | ○ | ○ | □ | □ |
| 2. | ○ | ○ | ○ | ○ | □ | □ |
| 1. | ○ | ○ | ○ | ○ | □ | □ |

Colours: (R)ed, (B)lue, (G)reen, (Y)ellow, (C)yan, (M)agenta

**Program: Factorial** (Filename: ComputeFactorial.java)

```
public class ComputeFactorial {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter number: ");
        int num = scanner.nextInt();
        int nFactorial = factorial(num);
        System.out.println(num + "! = " + nFactorial);
    }

    // Computes n factorial
    // Precondition: n >= 0
    public static int factorial(int n) {
        if (n == 0)
            return 1;
        else
            return n * factorial(n-1);
    }

}
```

**Program: North-east Paths** (Filename: NEPaths.java)

```
import java.util.*;

public class NEPaths {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter rows and columns apart: ");
        int rows = scanner.nextInt();
        int cols = scanner.nextInt();

        System.out.println("Number of North-east paths = "
                            + ne(rows,cols));
    }

    public static int ne(int x, int y) {

        // to be completed

    }

}
```

**Program: Tower of Hanoi** (Filename: TowerOfHanoi.java)

```
import java.util.*;

public class TowerOfHanoi {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter number of disks: ");
        int disks = scanner.nextInt();
        tower(disks, 'A', 'B', 'C');
    }

    // Towers of Hanoi
    // Precondition: n > 0
    public static void tower(int n,
              char source, char temp, char dest) {


        // to be completed


    }

}
```