Lab – 4

Week of Jan. 28, 2018

1 Instructions

1. Use the OCaml top-level to develop and debug your code.
2. You may assume that all inputs are valid unless otherwise stated in the problem.
3. In questions that require string outputs, be careful not to include leading or trailing whitespace.
4. You may submit and evaluate your code a *maximum* of 15 times without penalty. Subsequently, you will lose 2 marks per additional evaluation. Therefore, please ensure that you have thoroughly debugged your code before submitting.
5.

The following submission files are *required*:

1. `collatz.ml`
2. `josephus.ml`
3. `hanoi.ml`

2 Assignments

1. **Collatz Conjecture** (submission file: `collatz.ml`)

   The Collatz conjecture is one of the most famous problems in mathematics – it is very simple to understand, and every schoolboy who has heard of it has attempted to solve it. However, it has withstood all such attempts, and remains unsolved since it was posed in 1937. It is also known as the $3n + 1$ problem.

   Consider the following operation on an arbitrary positive integer:

   (a) If the number is even, divide it by two.

   (b) If the number is odd, triple it and add one.

Now form a sequence by performing this operation repeatedly, beginning with any positive integer, and taking the result at each step as the input at the next. The Collatz conjecture is: This process will eventually reach the number 1, regardless of which positive integer is chosen initially.

For instance, starting with n = 12, one gets the sequence 12, 6, 3, 10, 5, 16, 8, 4, 2, 1.For n = 19, it takes longer to reach 1: 19, 58, 29, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1.

(a) Write a *recursive* function `collatz: int -> int` that returns the number of steps to reach 1 when we start at an input integer. Examples:

i. `collatz 1 = 0`.

ii. `collatz 12 = 9`.

iii. `collatz 19 = 20`.

iv. `collatz 27 = 111`.

(b) Write a *recursive* function `max_collatz: int -> int` which takes in an integer n and returns the integer with *longest* sequence for any integer less than or equal to n. Examples:

i. `max_collatz 10 = 9`. Because `collatz 9 = 19` (the maximum, while all the other numbers equal to or below 10 have lower number of operations. For ex: `collatz 10 = 6, collatz 8 = 3,...`).

ii. `max_collatz 100 = 97`. Because `collatz 97 = 118` (the maximum of all numbers from 1 to 100).

2. **Josephus Problem** (submission file: `josephus.ml`)

People are standing in a circle waiting to be executed. Counting begins at a specified point in the circle and proceeds around the circle in a specified direction. After a specified number of people are skipped, the next person is executed. The procedure is repeated with the remaining people, starting with the next person, going in the same direction and skipping the same number of people, until only one person remains, and is freed. The problem: Given the number of people, starting point, direction, and number to be skipped, choose the position in the initial circle to avoid execution.

The problem gets its name from a Jewish historian, Josephus who was trapped with his 40 soldiers in a cave by the Romans. Instead of surrendering the jews decided to commit suicide instead. They stood in a circle and kill every 7th man. At the end, only Josephus and another soldier remained alive, and they chose to surrender instead of committing suicide.

Write a *recursive* function `josephus: int -> int -> int -> int` that takes 3 integer arguments: i) n, the number of people in the circle, ii) k, which indicates that every kth man is killed (that is, you count up to $k-1$ and kill the next person), and, iii) x, the starting point (that is, you start counting from this person). Assume that the positions are numbered $1, 2...n$ and that you move *counter clockwise*. The function should return the position that is safe. Examples:

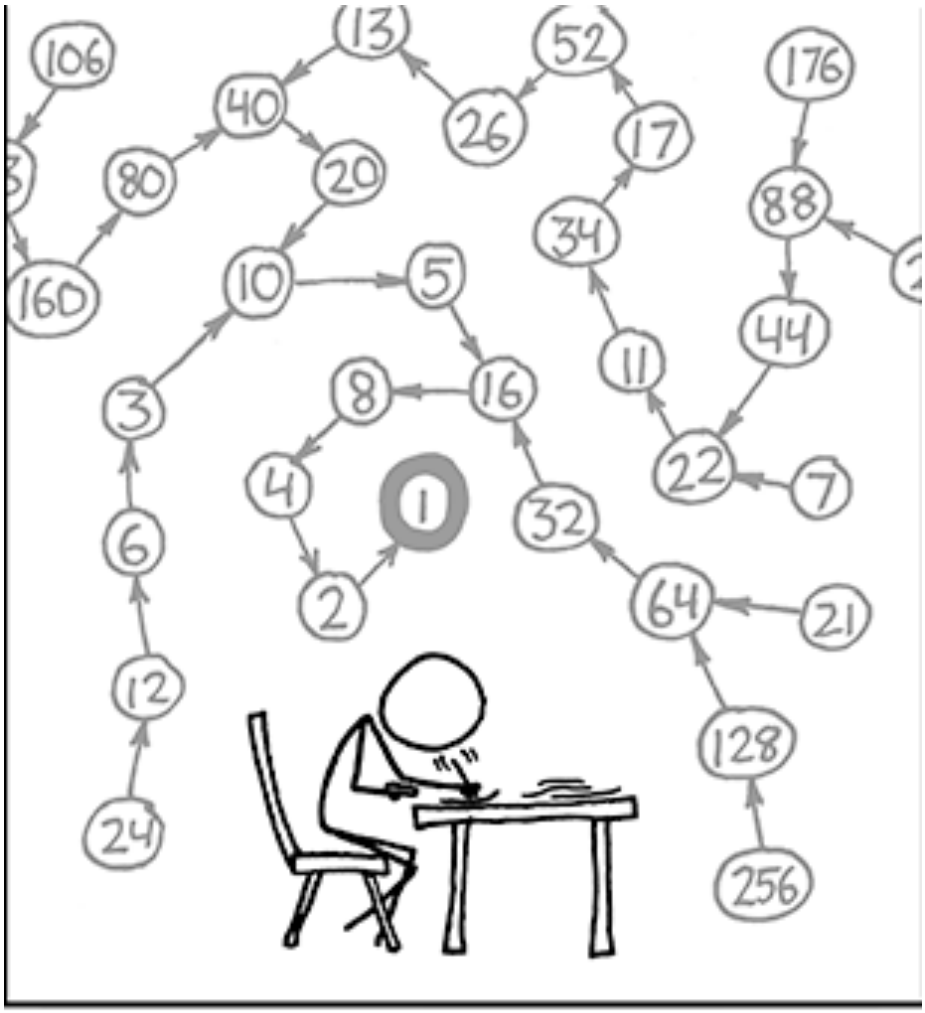

Figure 1: Relevant XKCD for the Collatz Conjecture

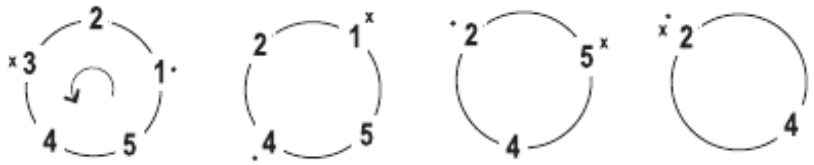

Figure 2: A simulation of the josephus problem: n = 5, k = 3 and starting position 1. Solution = 4

(a) `josephus 5 3 1 = 4`. See Figure 2 for explanation.

(b) `josephus 7 4 2 = 3`. The sequence of positions for killing will be : 5,2,7,6,1,4. Hence 3 will be the safe position.

3. **Tower of Hanoi** (submission file: `hanoi.ml`) The Tower of Hanoi, also known as the **Tower of Brahma** is a mathematical game or puzzle. It consists of three rods and a number of disks of different sizes, which can slide onto any rod. The puzzle starts with the disks in a neat stack in ascending order of size on one rod, the smallest at the top, thus making a conical shape (refer to Figure 3).

   The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

   (a) Only one disk can be moved at a time (called a "move").

   (b) A move consists of taking the upper-most disk from one of the stacks and placing it on top of another stack.

   (c) No disk may be placed on top of a smaller disk.

   There is a story about an Indian temple in Kashi Vishwanath which contains a large room with three time-worn posts in it, surrounded by 64 golden disks. Brahmin priests, acting out the command of an ancient prophecy, have been moving these disks in accordance with the immutable rules of Brahma since that time. The puzzle is therefore also known as the Tower of Brahma puzzle. According to the legend, when the last move of the puzzle is completed, the world will end.

   Assume that there are three towers 1, 2 and 3. The tower 1 consists of n disks (in sorted order), but 2 and 3 are empty. The task involves moving *all* n disks from 1 to 3.

   Write a *recursive* function `hanoi: int -> string` that takes in n, the number of disks and returns a string which shows the shortest sequence of moves to move all the disks from tower 1 to 3. Each line in the output represents a move – `(1, 2)` represents that a disk was moved from the top of Tower 1 onto Tower 2. Examples:

   (a) `hanoi 3` returns the following string (note the spaces and separate lines)
   `(1, 3)`

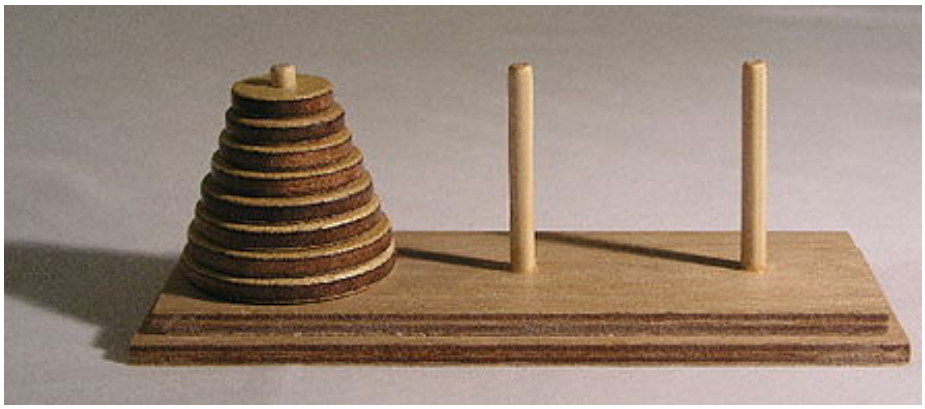

Figure 3: The initial state of the tower of Hanoi problem with 8 disks

```
(1, 2)
(3, 2)
(1, 3)
(2, 1)
(2, 3)
(1, 3)
```

(b) `hanoi 2` returns:

```
(1, 2)
(1, 3)
(2, 3)
```