1. Finite State Machines

Our treasure map, viewed more abstractly, is an example of a finite state machine (FSM, or sometimes called a finite state automaton, FSA):

A FSM is a model of a simple computer, that reads in a string of characters one by one, and might change its state depending on what character is read. A start state is specified. Also, an end state is specified; if for a certain input the FSM stops in the end state, we say the FSM accepts that input.

(a) The following FSM only allows numbers as inputs, and reads the digits left to right. What happens with the input 123? with 1243572? Which inputs does the FSM accept?

(b) Draw a FSM that accepts only words with exactly one vowel. You can assume the FSM only allows words as inputs and reads letters left to right, and you can let C stand for the set of all consonants and V for all vowels.

Challenge: A certain website requires a password. Letters, numbers, and some special symbols are allowed (like $, #, *, etc.), but at least one number and at least one symbol must be used. Draw a FSM that accepts valid passwords. You can let L be the set of letters, N the set of numbers (i.e. digits), and S the set of allowable special symbols.
2. Algorithms

(a) How would you define an algorithm? List some tasks that could be described by algorithms. How about a task that you don’t think could be described by an algorithm?

(b) Write down an algorithm for checking whether a given deck of cards includes the 5 of hearts.

(c) Now imagine we have a robot, Hal, that can recognize the value and suit of any playing card, can draw and discard cards from the deck, and can store true/false values in its memory. Let’s write some pseudocode, i.e. instructions for the robot to see if the 5 of hearts is in a given deck and report the answer back to us.
Wally, the farmer robot, was built to work in the henhouse, which consists of several stalls in a row, each containing a chicken. These are genetically modified chickens, which might lay more than one egg at a time. Wally starts at the far left of the henhouse, and knows how to do the following:

- move one stall to the left or right
- check if there is a stall immediately to his left or right
- count the number of eggs in the current stall
- simple math instructions, like addition, and comparing two numbers to determine which is greater or if they are equal
- store numbers and true/false values in his memory, and output stored quantities

(d) What does Wally do when he runs the following program?

```plaintext
numberOfSteps := 0
while (there is a stall to the right)
    move right
    numberOfSteps := numberOfSteps + 1
output numberOfSteps
```

(e) Write a program that has Wally output the total number of eggs in the henhouse.

(f) Write a program that has Wally output the largest number of eggs a chicken has laid. For example, if there are stalls with 2, 3, and 5 eggs, Wally should output 5.

**Challenge:** Write a program that has Wally output “true” if there are two nests right next to each other that both have eggs, and “false” otherwise.
Now let’s program another robot, Eva, to do some more interesting mathematics. Let’s suppose Eva already knows addition and subtraction, and understands statements with $>$, $<$, and $=$. She can store and output values, too, and can also read inputs we give her.

(g) What does the following program teach Eva to do?

```
(input x,y)
p := 0
while ( x > 0 )
    p := p + y
    x := x - 1
output p
```

(h) Write a program $\text{Remainder}(x,y)$ that outputs the remainder of $x$ divided by $y$. You can assume that both inputs are positive. *Hint: you only need subtraction.*

(i) Write a program $\text{DivisibleBy}(x,y)$ that outputs “true” if $x$ is divisible by $y$ and “false” otherwise. Note: you’ve already taught Eva how to find remainders.

**Challenge:** Write a program $\text{IsPrime}(x)$ that outputs “true” if $x$ is prime and “false” otherwise.
3. Is there anything an algorithm can’t do?

Eva 2.0 has gotten a major upgrade – a scanner. This allows her to read input given to her on a piece of paper.

(j) In both of the following programs, you can assume the input $x$ scanned is a positive number. Also suppose that if we tell Eva to stop, she does. What do these programs do?

```
x := input scanned
while (x > 0)
    if (IsPrime(x))
        output x
        stop
    x := x + 1
```

```
x := input scanned
while (x > 0)
    if (IsPrime(x))
        output x
    x := x + 1
```

There is a major difference between these programs. What is it?

(k) Now, let’s have Eva 2.0 do a little soul-searching. If she can scan any input on a piece of paper, we could input a number, a word, a paragraph…or a program. For example:

```
line := first line of input scanned
while (true)
    if (line contains the word while)
        output true
        stop
    if (line is the last line of the program)
        output false
        stop
    line := next line of input
```

Eva would output “true” if she scanned either of the programs above. Have we seen any programs that would lead to the output “false”?

Let’s call the program above $W$ (for “while”). What happens if we give the program $W$ as an input to the program $W$?
(l) Suppose you have a program $P$ on a piece of paper. We don’t know what $P$ does, but we know that it always outputs either “true” or “false”. Write another program $P'$ that always outputs the opposite of what $P$ outputs.

Now write a similar program $P''$ that:

- loops forever whenever $P$ outputs “true”, and
- stops whenever $P$ outputs “false.”

**Challenge:** Recall that above, we didn’t know what program $P$ actually did. So suppose that the program $P$ teaches Eva 2.0 to decide, given any program $I$ as input, whether she would eventually stop if she ran $I$; she should output true if yes, or false if no.

What happens when Eva runs program $P$ with an input of program $W$? Let’s call the two programs about prime numbers on the previous page $Q$ and $R$. What happens when Eva runs program $P$ with an input of program $Q$? with an input of $R$?

Now, what specifically does your program $P''$ do in terms of the input $I$?

What happens if you give the program $P''$ as an input to the program $P''$?

Is there anything an algorithm can’t do?