Unit 5 Exam: Computation Everywhere

QUESTION 1: Recursive Functions

For the following functions, (1) define the base case when y=0 [**a, c**] and x=0 [**b**] and (2) define the relationship between y and y+1 [eg. y+1 = what function or operation of y? for **[a,c,d,e]** and between x and x+1 [**b**]. You can assume you know how to find the values on the right side of each equation, so the question is specifically how to increment these operations for y+1 and x+1.

(a) $\exp(x, y) = x^y$

Base case: $y = 0$ then $exp(x,0) = x^0 = 1$

Recursive case: $exp(x, y + 1) = x^{y+1} = x \cdot x^y = mult(x, exp(x, y))$

$=> exp(x, y + 1) = mult(x, exp(x, y)) = x \cdot exp(x, y)$

(b) pred(x) = x - 1

Base case: $pred(0) = 0 - 1 = 0$ **(Note: negative values are returned as 0)**

Recursive case: $pred(x + 1) = (x + 1) - 1 = (x - 1) + 1 = add(pred(x),1)) = x$

$=> pred(x + 1) = add(pred(x),1)) = pred(x) + 1$

(c) sub(x, y) = x - y

Base case: $sub(x,0) = x - 0 = x$

Recursive case: $sub(x, y + 1) = x - (y + 1) = x - y - 1 = add(sub(x, y), -1)$

$=> sub(x, y + 1) = add(sub(x, y), -1) = sub(x, y) - 1$

(d) min(x, y) (**Note:** usual x+y is used in place of the add(x, y) for simplicity)

Define: $min(x, y) = (x + y) - max(x, y)$

See definition of max(x, y) in (e) below

Base case: $min(x,0) = (x + 0) - max(x,0) = x - x = 0$

Recursive case: $min(x, y + 1) = x + y + 1 - max(x, y + 1)$

$= x$ (**if** $y + 1 > x$)

$= y + 1$ (**if** $y + 1 = x$ **or** $x > y + 1$)

(e) max(x, y) (**Note:** usual x+y is used in place of the add(x, y) for simplicity)

Define: $max(x, y) = (x - y) + y$ (**Note: if** $y > x$ **then** $(x - y) = 0$ **and** $max(x, y) = y$)

Base case: $max(x,0) = (x - 0) + 0 = x$ (**Note:** this will be zero if $x < 0$ **or** $x = 0$)

Recursive case: $max(x, y + 1) = (x - (y + 1)) + (y + 1)$

$= (y + 1)$ (**if** $y + 1 > x$)

$= x$ (**if** $y + 1 = x$ **or** $x > y + 1$)

You may use the functions defined within the lecture as well as any functions defined here within your solutions:

add(x, y) = $x + y$

mult(x, y) = $x * y$

NOTE: Do not worry about negative values; assume any negative solutions return zero.

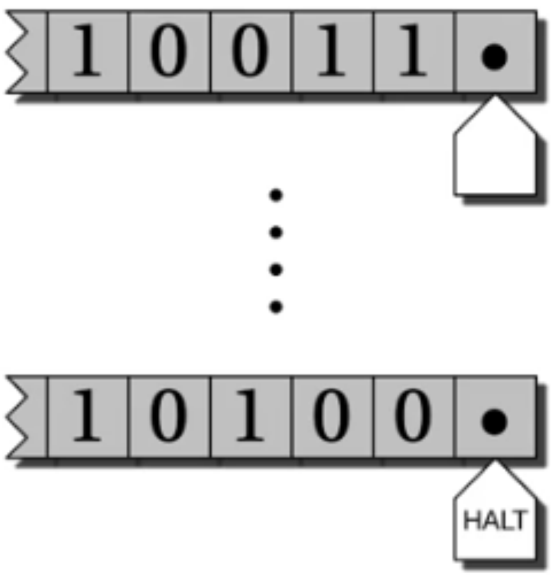
QUESTION 2: Turing Machines

Design a Turing machine that uses a head alphabet {•,0,1} to calculate the successor function (x+1). The machine starts in the state shown in the top depiction and should end in the state shown in the bottom depiction in the diagram below.

(1) Define the functions CARRY, RETURN, HALT and (2) describe the sequence of how they would be implemented by your machine when calculating the successor function on the head in the diagram. CARRY and RETURN function should be defined for the three possible conditions that the machine might read on the head {•,0,1}. The HALT function needs only to be defined for the decimal • state. You can assume that a "move" function exists and that negative move values indicate

leftward movement and positive move values indicate rightward movement, *eg*. "–3" would move the cursor 3 indices to the left.

When defining the functions, you may use a mathematic notation or describe these in natural language, *eg*. "If in state •, RETURN does..." or "*F* (RETURN, •) = ..."



A reminder about counting in binary: The integers shown in the question are 19 (top) and 20 (bottom). Binary increments in the following way: 0 + 1 = 1, 1 + 1 = 0 with the 1 carried to the next position (to the left). For example, the successor of 001 is 010, the successor of 010 is 011, and the successor of 011 is 100.

State Table for a Binary Successor Turing Machine

State	Symbol Read	Write Instruction	Move Instruction	Next State
Return	•	Write •	Move head left	Carry
	0	Write 0	Move head right	Return
	1	Write 1	Move head right	Return
Carry	•	Write 1	Move head right	Halt
	0	Write 1	Move head left	Halt
	1	Write 0	Move head left	Carry
Halt	•	Write •	No Move	Halt
	0	Write 0	Move head right	Halt
	1	Write 1	Move head right	Halt

Using the State Table above the following transformations computes the binary successor of 10011 · which is 10100 ·

Note: The machine starts in the Return State

10011 · *(read • write • move head left -> new State Carry)*

10011 · *(read 1 write 0 move head left -> new State Carry)*

10010 · *(read 1 write 0 move head left -> new State Carry)*

10000 · *(read 0 write 1 move head left -> new State Halt)*

10100 · *(read 0 write 0 move head right -> new State Halt)*

10100 · *(read 1 write 1 move head right -> new State Halt)*

10100 · *(read 0 write 0 move head right -> new State Halt)*

10100 · *(read 0 write 0 move head right -> new State Halt)*

10100 · *(read • write • no move -> new State **Halt**)*