CS 254: Computability and Complexity **Anonymous submission**
Problem Set #01 *October 17, 2019*

1. **A.**

```
; Binary addition - adds two binary numbers
; Input: two binary numbers, separated by a single space, eg '100 1110'

;Reads through first binary number and goes towards second
0 _ _ r 1
0 * * r 0

;brings us to least significant bit of second number if there is one
1 _ _ l 2
1 * * r 1

;replaces least significant bit of second number with a space and then depending on whether it w
2 0 _ l 3x
2 1 _ l 3y
2 _ _ l 7

;reads through the 2nd binary number if the least significant bit was 0
3x _ _ l 4x
3x * * l 3x

;reads through the 2nd binary number if the least significant bit was 1
3y _ _ l 4y
3y * * l 3y

;adds 0 to the first non-x/y bit in the first binary number. If the result is a 0 replaces with
4x 0 x r 0
4x 1 y r 0
4x _ x r 0
4x * * l 4x ; skip the x/y's

;adds 1 to the first non-x/y bit in the first binary number. If the result is a 1 replaces with
4y 0 1 * 5
4y 1 0 l 4y
4y _ 1 * 5
4y * * l 4y ; skip the x/y's

;moves through the first binary number until an empty space or an x or y is encountered
5 x x l 6
5 y y l 6
5 _ _ l 6
5 * * r 5

; replaces the right most digit of the first binary number with an x or y then moves to state 0
6 0 x r 0
6 1 y r 0
```

```
; convert all x's and y's to 0's and 1's respectively by sweeping to the left ending with the m
7 x 0 l 7
7 y 1 l 7
7 _ _ r halt
7 * * l 7
```

B.

The Turing machines reads through the whole string until it finds the blank space after the 2nd binary number. It then replaces the least significant bit of the 2nd number with an empty space and adds it to the first digit, non-x/y, bit of the first binary number. Then goes moves back to the space between the two numbers and then converts the right most digit, non-x/y, bit of the first number to either an x or y if it is a 0 or 1 respectively. Then we repeat the process. Once there are no numbers in the 2nd binary number we go through the first binary number replacing the xs and ys with 0s and 1s respectively.

C.

```
;All the plus states move us right and mark the end of the string with a z
8+ _ _ r 9+
8+ * * r 8+
9+ _ _ r 9+
9+ * * r 10+
10+ _ z l 11
10+ * * r 10+

; move all the way to the least significant bit of the 3rd number
8 _ _ r 9
8 * * r 8
9 _ _ r 9
9 * * r 10
10 _ _ l 11
10 * * r 10

11 z _ l halt-accept
11 0 _ l 12 ; it's a 0 we need to check
11 1 _ l 12+ ; it's a 1 we need to check

; handle the case where 0 should be the least significant bit
12 _ _ l 13
12 * * l 12
13 1 _ r halt-reject
13 0 _ r 8
13 _ _ l 13

; handle the case where 1 should be the least significant bit
12+ _ _ l 13+
12+ * * l 12+
13+ 1 _ r 8
13+ _ _ l 13+
13+ 0 0 r halt-reject
```

```
; Sums first two numbers
0 _ _ r 1
0 * * r 0
1 _ _ l 2
1 * * r 1
2 0 _ l 3x
2 1 _ l 3y
2 _ _ l 7
3x _ _ l 4x
3x * * l 3x
3y _ _ l 4y
3y * * l 3y
4x 0 x r 0
4x 1 y r 0
4x _ x r 0
4x * * l 4x ; skip the x/y's
4y 0 1 * 5
4y 1 0 l 4y
4y _ 1 * 5
4y * * l 4y ; skip the x/y's
5 x x l 6
5 y y l 6
5 _ _ l 6
5 * * r 5
6 0 x r 0
6 1 y r 0
7 x 0 l 7
7 y 1 l 7
7 _ _ r 8+
7 * * l 7
```