PHIL 379
Assignment 2

Safian Omar Qureshi
ID 10086638

1. We can show this in two different but equally applicable ways:

i) Any two place function f can be related with a language $L_f = \{ (x,y) \mid y = f(x,y) \}$. Then we can discuss the decidability of L_f to show non Turing computability of function f.

Consider the language $T_D = \{ \langle M \rangle \mid \langle M \rangle \notin L(M) \}$ as a counter example. Assume T_D is decidable and let M_D be its decider. Then consider the case where M_D accepts $\langle M_D \rangle$. But then we have $\langle M_D \rangle \in L(M_D)$ so $\langle M \rangle \notin T_D$ which contradicts with our defined language.

This shows that our counter example is undecidable and so the two place function f relating to T_D is non Turing computable.

ii) We can also give a more direct example in the form of a diagonal function d with an added second 'dummy' argument.

Let $f_1, f_2, f_3, \ldots f_n$ be a listing of all 2 argument Turing computable functions. Let d be a 2 place function (and assume it is Turing computable) such that $d(n,y) = 1$ if the n^{th} function in the list of Turing computable functions is defined and returns 1, otherwise $d(n,y) = 0$. This can be also said in the following way

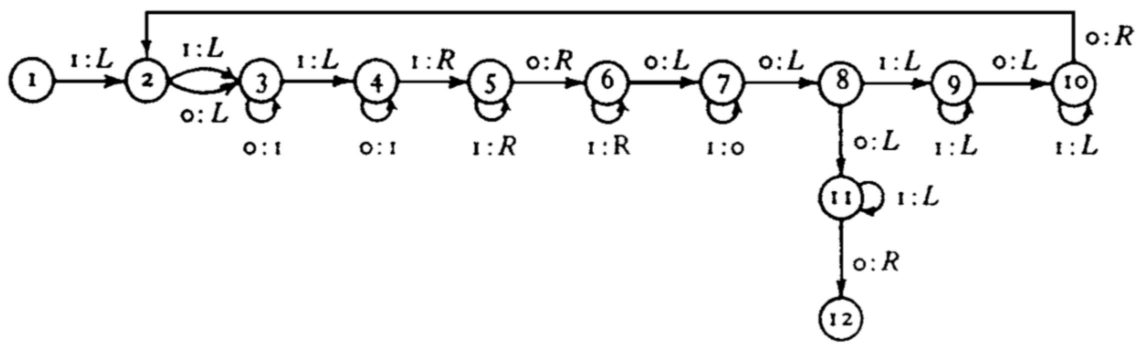
$$d(n,y) = \begin{cases} 1 \text{ if } f_n(n,y) \text{ is defined and } = 1 \\ 0 \text{ otherwise} \end{cases}$$

Assume d is the m^{th} Turing computable function and somewhere on the listing. Then for each positive integer n, either $d(n,y)$ and $f_m(n,y)$ are both defined and equal or neither of them is defined. But we have

$$f_m(m,y) = d(m,y) = \begin{cases} 1 \text{ if } f_m(m,y) \text{ is defined and } = 1 \\ 0 \text{ otherwise} \end{cases}$$

which leads to the contradiction: either $f_m(m,y)$ is undefined in which case $n = m$ tells us that it is defined and has a value of 1. Or $f_m(m,y)$ is defined and has a value of != 1 in which case $n = m$ tells us it has value of 1. Since we have shown a contradiction from the assumption that d does appear somewhere in the listing, the supposition is false and so d is not a two place Turing computable function.

2. The following flowchart represents our Turing machine[1];



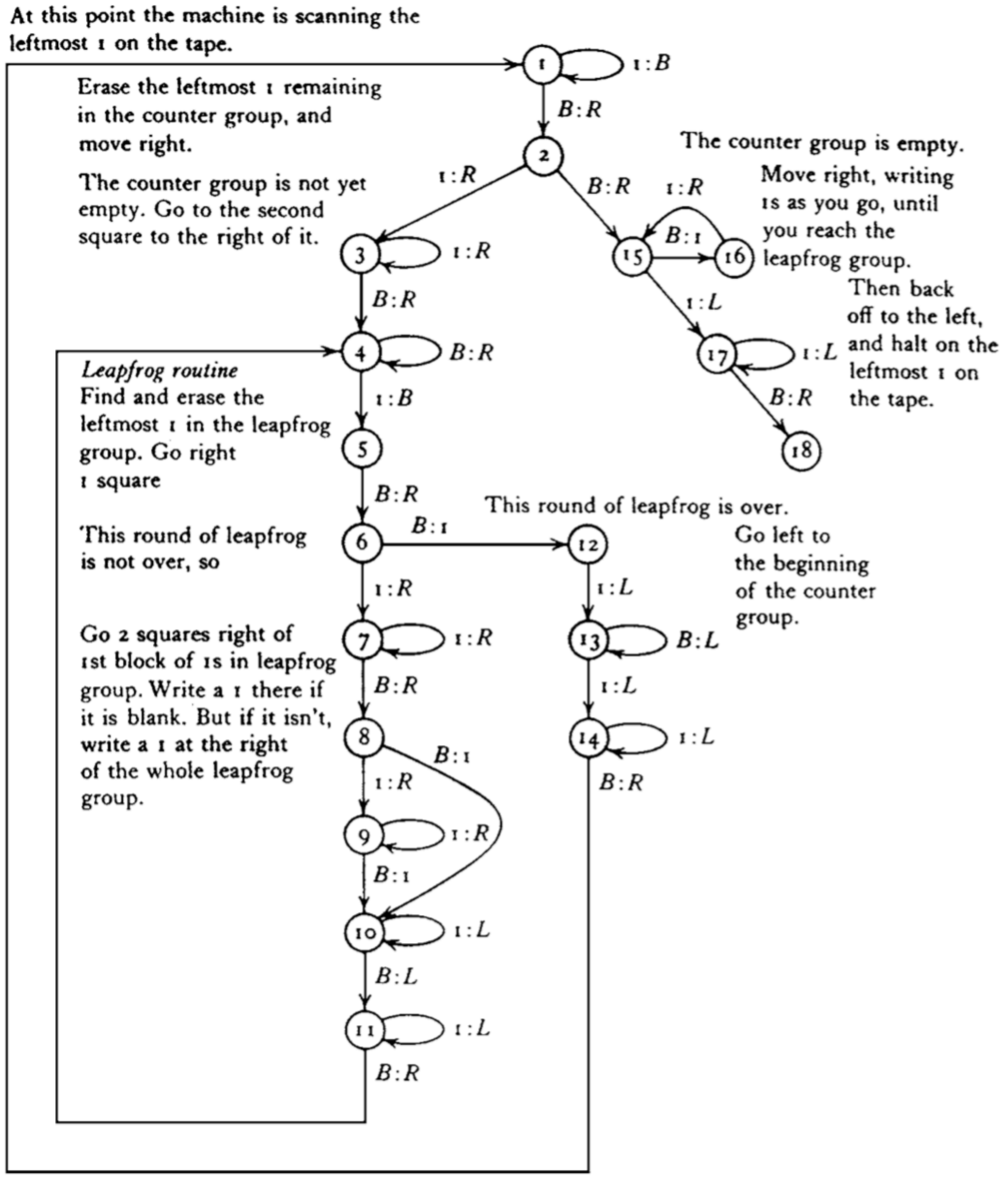
The set of configurations it follows are:

1_1 1 1 1	t = 0	1 1 $0_5$1 1 1 1	t = 7	1 1 0 1 1 1 0_7	t = 14
$0_2$1 1 1 1	t = 1	1 1 0 $1_6$1 1 1	t = 8	1 1 0 1 1 $1_8$0	t = 15
$0_3$0 1 1 1 1	t = 2	1 1 0 1 $1_6$1 1	t = 9	1 1 0 1 $1_9$1	t = 16
$1_3$0 1 1 1	t = 3	1 1 0 1 1 $1_6$1	t = 10	1 1 0 $1_9$1 1	t = 17
$0_4$1 0 1 1 1 1	t = 4	1 1 0 1 1 1 1_6	t = 11	1 1 $0_9$1 1 1	t = 18
$1_4$1 0 1 1 1 1	t = 5	1 1 0 1 1 1 1 0_6	t = 12	1 1_{10}0 1 1 1	t = 19
1 $1_5$0 1 1 1 1	t = 6	1 1 0 1 1 1 1_7	t = 13	1_{10}1 0 1 1 1	t = 20
				0_{10}1 1 0 1 1 1	t = 21
				0 $1_2$1 0 1 1 1	t = 22

So at time 22, we have configuration 0 $1_2$1 0 1 1 1

[1] Boolos, G. S., Burgess, J. P., & Jeffrey, R. C. (2010). *Computability and logic*. pg 28. Cambridge: Cambridge Univ. Press.

3. The following flowchart represents our Turing machine[2];



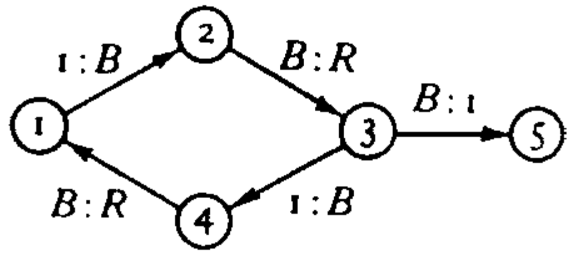
For the computation of g(3,2) where the first argument is 3 and the second argument is 2, the initial configuration will look like $1_1$1 1 B 1 1. The final configuration will be 1 1 1 1 1, or more specifically 1_n1 1 1 1 where n is the nth state which there is no instruction on what to do so the machine will be halted at that point[3].

---

[2] Boolos, G. S., Burgess, J. P., & Jeffrey, R. C. (2010). *Computability and logic*. pg 30. Cambridge: Cambridge Univ. Press.

[3] Boolos, G. S., Burgess, J. P., & Jeffrey, R. C. (2010). *Computability and logic*. pg 32. Cambridge: Cambridge Univ. Press.

4. The following flowchart represents our Turing machine[4]



which essentially has the task of counting the amount of 1s in a given block of unbroken 1s. To end the count, the machine must encounter either a 0 if the initial amount of 1s in the block is even, or encounter a 0 followed by a 1 if the initial amount of 1s is odd. I.e

Case 1: tape fed is 1111. The Turing machine stops at 01111 encountering a leftmost 0, given that the unbroken block of 1s contained an even amount (4) 1s.

Case 2: tape fed is 11111. The Turing machine stops at 1011111 encountering leftmost 0 followed by 1, given that the unbroken block of 1s contained an odd amount (5) 1s.

To show that there is a Turing machine that, if it computes a two-place function g(x,y), then there must also be a machine that computes the one-place function f, where f(x) = g(x,x), we can start by supposing that one such machine exists initially.

Suppose there exists a Turing machine T_1 that computes a two-place function *g(x,y)*. It follows that for a given x = y, the two place function *g(x,x)* is computable, given our supposition. Then, the two place function *g(x,y)*, at a given x = y, is effectively a function of one argument when it is *g(x,x)*. We can let this function be represented by a one-place function *f(x) = g(x,x)*. Since g(x,x) is computable, this in turn means the function *f(x)* is indeed computable as well by some Turing machine T_2.

[4] Boolos, G. S., Burgess, J. P., & Jeffrey, R. C. (2010). *Computability and logic*. pg 29. Cambridge: Cambridge Univ. Press.