CIS 3223 Homework 5 | **Name: Parth Patel**

Dr Anthony Hughes | **Temple ID (last 4 digits): 5761**

Simple non-graphing calculator | **Make: None**

1 (6 pts) Determine if N is a perfect square (complete table).

(a) N = 371 — 371 is a perfect square **true** **false**

L	U	U - L	U - L > 1	M = ⌊ L + U)/2⌋	M^2	Action
1	371	370	True	186	34596	U = M
1	186	185	True	93	8,649	U = M
1	93	92	True	47	2,209	U=M
1	47	46	True	24	576	U=M
1	24	23	True	12	144	L = M
12	24	12	True	18	324	L = M
18	24	6	True	21	441	U = M
18	21	3	True	19	361	L = M
19	21	2	True	20	400	U = M
19	20	1	False			

(b) N = 324 — 324 is a perfect square **true** **false**

L	U	U - L	U - L > 1	M = ⌊ L + U)/2⌋	M^2	Action
1	324	323	T	163	26,569	U = M
1	163	162	T	82	6,724	U = M
1	82	81	T	41	1,681	U = M
1	41	40	T	21	441	U = M
1	21	20	T	11	121	L=M
11	21	10	T	15	225	L = M
15	21	6	T	18	324	END

2 (**extra credit, 3 pts**) 2.17. (P74)

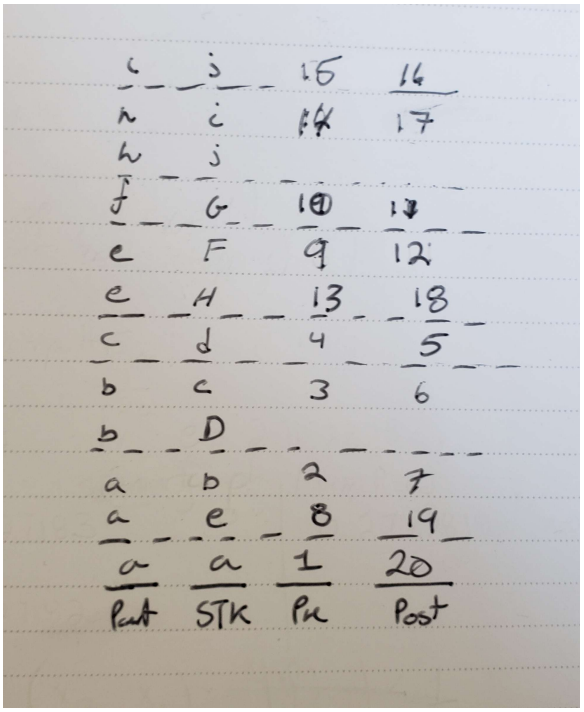
```
function idx = idx_eq_val(arr)
    L = 1; %left pointer
    R = length(arr); %right pointer

    while L < R

        %check end points
        if arr(L) == L
           break
        end

        if arr(R) == R
           L = R;
           break
        end

        %find midpoint
        M = floor((L + R) / 2);

        if arr(M) > M
           R = M - 1;
        elseif arr(M) < M
           L = M + 1;
        else
            % we found the value for idx in the middle
            L = M;
            break;
        end
    end

    %see why loop was broken
    if arr(L) == L
        idx = L;
    else
        idx = -1;
    end
end
```

3 (7 pts) Perform a dfs on the following undirected graph $G = (V, E)$ starting at vertex A; use the ordering given in the adjacency list representing E.

Push neighbors onto the stack in reverse order.

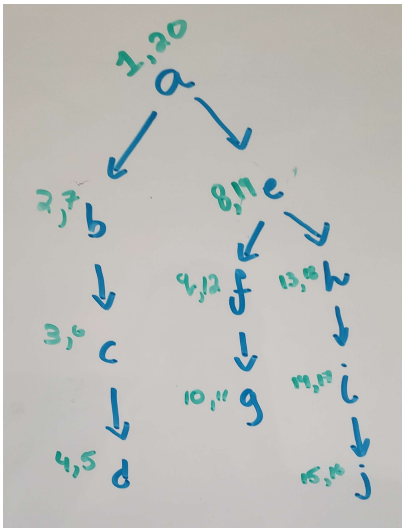
V = {A, B, C, D, E, F, G, H, I, J}

E{A} = [B, E]
E{B} = [A, C, D]
E{C} = [B, D]
E{D} = [C, B]
E{E} = [A, F, H]
E{F} = [E, G]
E{G} = [F]
E{H} = [E, I, J]
E{I} = [H, J]
E{J} = [H, I]



parent	a	a	b	c	a	e	f	e	h	i
pre	1	2	3	4	8	9	10	13	14	15
post	20	7	6	5	19	12	12	18	17	16
vertex	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**

Draw the resulting spanning tree of K . Add pre/post numbers.

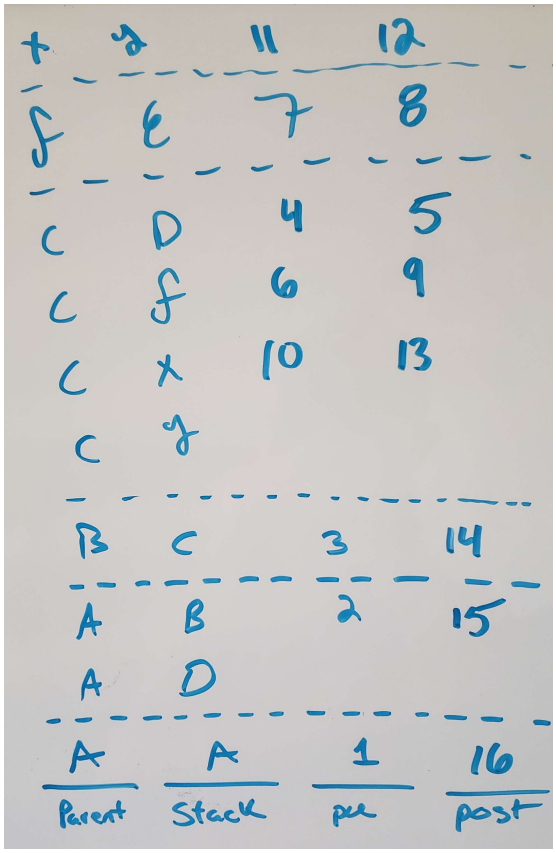


4 (7 pts) Perform a `dfs`(pre/post) on the directed graph $G = (V, E)$ starting at vertex A; use the ordering given in the adjacency list representing E.

Push neighbors onto the stack in reverse order.

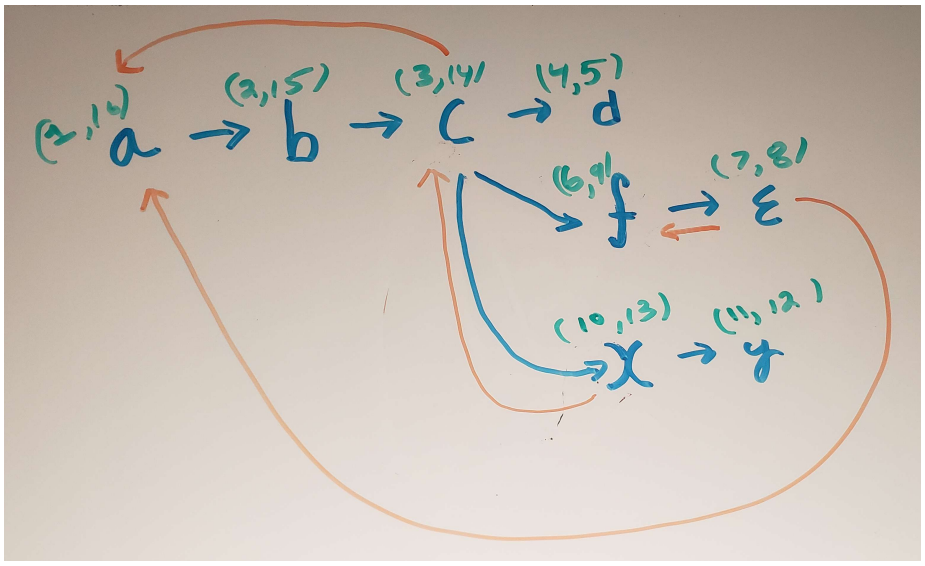
V = {A, B, C, D, E, F, X, Y}

E{A} = [B, D]
E{B} = [C]
E{C} = [A, D, F, X, Y]
E{D} = [ ]
E{E} = [A, F]
E{F} = [D, E]
E{X} = [C, Y]
E{Y} = [F]



parent	a	a	b	c	f	c	c	x
pre	1	2	3	4	7	6	10	11
post	16	15	14	5	8	9	13	12
vertex	A	B	C	D	E	F	X	Y

Draw a spanning tree (horizontally) consisting of the **tree edges**. Add pre/post numbers and any **back edges**.



Orange connections are **back-edges.**

List of Back Edges:

1. **(x, c)**
2. **(e, a)**
3. **(e, f)**
4. **(c, a)**