1. (8 points)Consider the weighted graph below:

(a) Demonstrate Prim's algorithm starting from vertex A. Write the edges in the order they were added
to the minimum spanning tree. s = a, (a,e)w=4, (e,b)w=3, (b,c)w=6, (c,d) w=1, (c,g)w=9, (G,f)w=7,(d,h)w=12

(b) If each edge weight is increased by 1 will this change the
minimum spanning tree? Explain. no, it will not change the spanning tree because each edge increased the same amount meaning that nothing will change on the mst because when comparing each edge the difference between them would increase the same so the mst will remain the exact same

2(7 points)
A region contains a number of towns connected by roads. Each road is labeled by the average number of minutes required for a fire engine to travel to it. Each intersection is labeled with a circle. Suppose that you work for a city that has decided to place a fire station at location G. (While this problem is small, you want to devise a method to solve much larger problems).

(a) What algorithm would you recommend be used to find the fastest route from the fire station to
each of the intersections? Demonstrate how it would work on the example above if the fire station is
placed at G. Show the resulting routes and times.
Dijkstra's algorithm would be the best choice for finding the fastest routes from one station to each of the intersections.

	S	d_v	p_v
A	T	12	C
B	T	6	H
C	T	8	D
D	T	5	E
E	T	2	G
F	T	8	G
G	T	0	-
H	T	3	G

(b) Suppose one ”optimal” location (maybe instead of G) must be selected for the fire station such that it minimizes the time to the farthest intersection. Devise an algorithm to solve this problem given an arbitrary road map. Analyze the running time complexity of your algorithm when there are f possible locations for the fire station (which must be at one of the intersections) and r possible roads.

```
function FireStationLoco(G,w)
        d= array[1… sizeof(G.V)]
        p= array[1…sizeof(G.V)]
        minDist= INF
        for v in G.V
                d = Dijkstra(G, w, v)
                p= Dijkstra(G, w, v)
                if(max(d) < minDist)
                        minDist=max(d)
        return (minDist, d, p)
function Dijkstra(G, w, s)
        d= array[1… sizeof(G.V)]
        p= array[1…sizeof(G.V)]
        for v in G.V
                d[v] = INF
                p[v] = NULL
        d[s] = 0
        Queue = G.V
        while Queue not empty
                u = ExtractMin(Queue)
                for v in Queue and in u.Neighbors
                        if d[u] + w[u][v] < d[v]
                                d[v] = d[u] + w[u][v]
                                p[v] = u
return (d,p)
```

The running time of the algorithm since we used arrays to hold our values will be $O(|V|^3)$ since we must try all vertices to get our optimal solution.

(c) In the above graph what is the “optimal” location to place the fire station? Explain.

To determine which location would be the best we would have to place the fire house at each location and since response time is the most important factor in saving lives. The spot with the lowest response time to the furthest location would be the most optimal location for the fire station. After doing the calculations of each location the most optimal was E the route that took the longest was 10 going E->d, d->c, c->a gave a weight of 10, and the next closest location was D with the best response time of 12(which it had 2 with that time).

3. (15 points)
Suppose there are two types of professional wrestlers: “Babyfaces” (“good guys”) and “Heels” (“bad guys”). Between any pair of professional wrestlers, there may or may not be a rivalry. Suppose we have n wrestlers and we have a list of r pairs of rivalries

(a) Give pseudocode for an efficient algorithm that determines whether it is possible to designate some of the wrestlers as Babyfaces and the remainder as Heels such that each rivalry is between a Babyface and a Heel. If it is possible to perform such a designation, your algorithm should produce it.

```
Wrestler(array of wrestlers, array of rivals)
    assign wrestler to the correct group
    while all wrestlers are not assigned
            for through the rivalry
                    verify if wrestler is a group
                    if in group assign other wrestler to other group
                    if in the same group return no
            check to see if changes were made then assign wreslter in the list who
is not in group
    print results
```

(b) What is the running time of your algorithm?
the running time of the pseudo code is O(N*R) which is the number of wrestlers times the rivalries