Q1: Heavy load of this problem lays in sorting. To solve the problem we need to sort them in such order that we minimize the weighted completion time. I used bubble sort, $O(n^2)$, for sorting jobs in ascending order according to their time/weight. I compared this solution to others :

Time/Weight, Ascending	1102
Just time, Ascending	1454
Just weight, Ascending	27786
Weight/Time, Ascending	27986

Q2:

a)

#	Month 1	Month 2	Month 3	Month 4
NY	1	2	1	2
SF	2	1	2	1

Let's say M is 10, given algorithm prints :

```
NY > SF > NY > SF
Cost : 1 + 1 + 1 + 1 + 4*10 = 44
```

If we didn't travel at all cost would be :

```
Cost : 1 + 2 + 1 + 2 = 6
```

This algorithm doesn't give an optimal solution as seen above. So if travel cost is too much and cost difference between cities isn't big enough it's not worth to travel between them.

b) Optimal plan either ends in NY or in SF. If it end in NY it will pay NY[n] plus

-optimal plan on n-1 months, ending NY

OR

-optimal plan on n-1 months, ending SF, plus the traveling cost

Algorithm :

```
cost1 = city1[0]
cost2 = city2[0]
for i in range (1, numOfMonths) :
      cost1 = city1[i] + min(cost1, travelCost + cost2)
      cost2 = city2[i] + min(cost2, travelCost + cost1)
```

Example :

```
      ny = [1,3,20,30]
      sf = [50,20,2,4]
      m = 10
      plan(sf,'SF',ny, 'NY', m, 4)  #(20, ['NY', 'NY', 'SF', 'SF'])
      la = [1,2,1,2]
      tx = [2,1,2,1]
      m = 5
     plan(la,'LA',tx, 'TX', m, 4)  #(6, ['LA', 'LA', 'LA', 'LA'])
```