Software Engineering Intern Examination

Problem 1:

Given two addresses in an unstructured format, measure the similarities between those addresses.

This is an open question. Feel free to interpret the problem and propose the solution.

Sample addresses:
- PERUM. PURI ARSANA JL. RAYA CIPUTAT PARUNG KM.11 E-12 IDN
- Jalan budi mulia gang F3 No.29D Rt.04 Rw.02, Jakarta Selatan. Indonesia.
- Level 1, Tower 6, Avenue 3, The Horizon, Bangsar South, No. 4, Jalan Kerinchi
- Jalan budi mulia Gg. F2 Nomor 29 D RT. 04 Rw.02, JakSel.
- Green garden M17A no.44 Jakarta Barat - 11520
- Jln budi mulia Gg. F2 Nomor 28 D RT. 04 Rw.02, JakSel. IDN.
- Jl BudiMulia Gang f2 Nomor 28 D RT. 04/02, Jakarta Selatan. IDN.

Problem 2:

```
transactions = [
 { id: 1, email: 'e1', phone: 'p1', card: 'c1' },
 { id: 2, email: 'e2', phone: 'p2', card: 'c2' },
 { id: 3, email: 'e1', phone: 'p3', card: 'c3' },
 { id: 4, email: 'e4', phone: 'p4', card: 'c4' },
]
```

A set of transactions can be concluded from same customer if those transactions used the same email, phone, or card.
With 4 transactions above, we can conclude that there are 3 customers:

```
customer1:
 transactions: [1,3]
 emails: [e1],
 phones: [p1, p3],
 cards: [c3, c3]
```

```
customer2:
  transactions: [2]
  emails: [e2],
  phones: [p2],
  cards: [c2]

customer3:
  transactions: [4]
  emails: [e4],
  phones: [p4],
  cards: [c4]
```

If we add new transaction ({ id: 5, email: 'e2', phone: 'p4', card: 'c5' }) to the transactions data, customer2 & customer3 will be merged into one customer because they used same email & phone.

```
customer1:
  transactions: [1,3]
  emails: [e1],
  phones: [p1, p3],
  cards: [c3, c3]

customer2:
  transactions: [2,4,5]
  emails: [e2, e4],
  phones: [p2, p4],
  cards: [c2, c4, c5]
```

New transactions data will keep coming.

Please create a solution (algorithm & data structure) to show all customer every time new transaction added to the transaction data.

Problem 3:

Create a phone number normalizer that will pass the all test cases bellow.

```
test_cases = [
 {phone: '-', normalized_phone: '-'},
 {phone: '0', normalized_phone: '0'},
 {phone: '62', normalized_phone: '62'},
 {phone: '(null)', normalized_phone: '(null)'},
 {phone: '+6281298490805', normalized_phone: '6281298490805'},
 {phone: '6281298490805', normalized_phone: '6281298490805'},
 {phone: '08119284411', normalized_phone: '628119284411'},
 {phone: '+1 (804) 244-3470', normalized_phone: '18042443470'},
 {phone: '*083831397998', normalized_phone: '6283831397998'},
 {phone: '+1408-888-4919', normalized_phone: '14088884919'},
 {phone: '+1 917 856 9984', normalized_phone: '19178569984'},
 {phone: '?+62 822 429737527', normalized_phone: '6282242973752'},
 {phone: '646.490.2691', normalized_phone: '6464902691'},
 {phone: '+626281322522898', normalized_phone: '626281322522898'},
 {phone: '+852-92730944', normalized_phone: '85292730944'},
 {phone: '+62-081377229637', normalized_phone: '6281377229637'},
 {phone: '82664848155', normalized_phone: '82664848155'},
 {phone: '08111338062 / 08788', normalized_phone: '6281113380620878 8'},
 {phone: '(021) 5736789', normalized_phone: '62215736789'}
]
```

Problem 4:

There is a building that has five floors, and there are six people at the same time want to use the elevator, detail of their origin and destination floor listed below.

```
p1                                   =                                   {
 orig:                                                                  1,
 dest:                                                                  5,
}
```

```
p2                                          =                                          {
 orig:                                                                                 2,
 dest:                                                                                 3,
}

p3                                          =                                          {
 orig:                                                                                 2,
 dest:                                                                                 4,
}

p4                                          =                                          {
 orig:                                                                                 3,
 dest:                                                                                 4
}

p5                                          =                                          {
 orig:                                                                                 3,
 dest:                                                                                 1
}

ps6                                          =                                         {
 orig:                                                                                 5,
 dest:                                                                                 1
}
```

-----
initial elevator position is on the 1st floor.

Cost for every movement:

- move 1 floor up/down = 1 point
- stop & embark/disembark passenger = 2

Please create a function that will cost the lowest possible point and order the people from who will be embarked and disembarked first. Means there are two lists, first sorts by who embark first and second disembark first.

Problem 5:

Given a list of bus schedule that contains arrival and depature, determine maximum number of bus inside the station at one point of time.

Example:

A = arrival D = departure

Bus 1 A 10:00 D 10:05

Bus 2 A 10:05 D 10:15

Bus 3 A 10:10 D 10:30

Bus 4 A 10:25 D 10:40

Bus 5 A 10:45 D 10:50

The answer must

1. give correct result and passes test
2. provide the big O

Problem 6:

There are two different strings, that have some same chars in common.

please create a function that counts the shortest number of possible char need to be removed in both strings.

to make them become a perfect anagram, which is both strings have exactly the same chars.

Example:

```
string1 = abcde
string2 = bczah
ooutput = 4 # removing z,h,d,e
```

Problem 7:

We need to create an app for storing customer name. this app will contain two main functions:

1. add the customer name,

2. find a customer name by given string (single or some char) and should return the number of customers that have name start with given string.

Spec:

- Create a CLI program that will receive n as the first requested input, where isn the number of how many operations that can be performed in one program execution.
- Add function will be triggered by giving an input start with add prefix example: add robbeth it should add robbeth as the customer name, and no need to print any into stdout
- Find function will be triggered by giving an input start with prefixfind. example: find rob it should count how many customers' name start with prefix rob by printing only the integer number into stdout.

Example:

Boilerplate code in ruby:

note : *you can choose any language you like*

```
n = gets.strip.to_i

for i in 0..(n-1)
 func,string = gets.strip.split(" ")
 if func == 'add'
 # do add operation
 elsif func == 'find'
 # do find operation
 end
end
```

Given input & expected output in order:

```
input  : input : 10
input  : add hendra
input  : add sreix
input  : add maji
input  : add srimulyani
input  : add hendro
input  : add heru
input  : find sr
output : 2 # because there are 2 names started with `sr`
input  :  find he
output : 3 # because there are 3 names started with `he`
```

```
input  :  find hen
output : 2 # because there are 2 names started with `hen`
input  :  find m
output : 1 # because only 1 names started with `ma`
```