WeHelp

Assignment - Week 2

Complete tasks by Python and JavaScript without third party libraries.

Note: Python code should run on Python 3 or above version.

Task 1:

We just received messages from 5 friends in JSON format, and we want to take the green line, including Xiaobitan station, of Taipei MRT to meet one of them. Write code to find out the nearest friend and print name, based on any given station currently located at and station count between two stations.

Note: Never change existing code.

Python

```
def find_and_print(messages, current_station):
        # your code here
messages={
        "Leslie":"I'm at home near Xiaobitan station.",
        "Bob":"I'm at Ximen MRT station.",
        "Mary":"I have a drink near Jingmei MRT station.",
        "Copper":"I just saw a concert at Taipei Arena.",
        "Vivian":"I'm at Xindian station waiting for you."
}
find_and_print(messages, "Wanlong") # print Mary
find_and_print(messages, "Songshan") # print Copper
find_and_print(messages, "Qizhang") # print Leslie
find_and_print(messages, "Ximen") # print Bob
find_and_print(messages, "Xindian City Hall") # print Vivian
```

JavaScript

```
function findAndPrint(messages, currentStation){
        // your code here
}
const messages={
        "Bob":"I'm at Ximen MRT station.",
        "Mary":"I have a drink near Jingmei MRT station.",
        "Copper":"I just saw a concert at Taipei Arena.",
        "Leslie":"I'm at home near Xiaobitan station.",
        "Vivian":"I'm at Xindian station waiting for you."
};
findAndPrint(messages, "Wanlong"); // print Mary
findAndPrint(messages, "Songshan"); // print Copper
findAndPrint(messages, "Qizhang"); // print Leslie
findAndPrint(messages, "Ximen"); // print Bob
findAndPrint(messages, "Xindian City Hall"); // print Vivian
```

Task 2:

Assume we have consultants for consulting services. Help people book the best matching consultant in a day, based on hours, service durations, and selection criteria.

1. Booking requests are one by one, order matters.
2. A consultant is only available if there is no overlapping between already booked time and an incoming request time.
3. If the criteria is "price", choose the available consultant with the lowest price.
4. If the criteria is "rate", choose the available consultant with the highest rate.
5. If every consultant is unavailable, print "No Service".

Note: Never change existing code.

Python

```
# your code here, maybe
def book(consultants, hour, duration, criteria):
        # your code here
consultants=[
        {"name":"John", "rate":4.5, "price":1000},
        {"name":"Bob", "rate":3, "price":1200},
        {"name":"Jenny", "rate":3.8, "price":800}
]
book(consultants, 15, 1, "price") # Jenny
book(consultants, 11, 2, "price") # Jenny
book(consultants, 10, 2, "price") # John
book(consultants, 20, 2, "rate") # John
book(consultants, 11, 1, "rate") # Bob
book(consultants, 11, 2, "rate") # No Service
book(consultants, 14, 3, "price") # John
```

JavaScript

```
// your code here, maybe
function book(consultants, hour, duration, criteria){
        // your code here
}
const consultants=[
        {"name":"John", "rate":4.5, "price":1000},
        {"name":"Bob", "rate":3, "price":1200},
        {"name":"Jenny", "rate":3.8, "price":800}
];
book(consultants, 15, 1, "price"); // Jenny
book(consultants, 11, 2, "price"); // Jenny
book(consultants, 10, 2, "price"); // John
book(consultants, 20, 2, "rate"); // John
book(consultants, 11, 1, "rate"); // Bob
book(consultants, 11, 2, "rate"); // No Service
book(consultants, 14, 3, "price"); // John
```

Task 3:

Find out whose middle name is unique among all the names, and print it. You can assume every input is a Chinese name with 2 ~ 5 words. If there are only 2 words in a name, the middle name is defined as the second word. If there are 4 words in a name, the middle name is defined as the third word.

Note: Never change existing code.

Python

```
def func(*data):
        # your code here
func("彭大牆", "陳王明雅", "吳明") # print 彭大牆
func("郭靜雅", "王立強", "郭林靜宜", "郭立恆", "林花花") # print 林花花
func("郭宣雅", "林靜宜", "郭宣恆", "林靜花") # print 沒有
func("郭宣雅", "夏曼藍波安", "郭宣恆") # print 夏曼藍波安
```

JavaScript

```
function func(...data){
        // your code here
}
func("彭大牆", "陳王明雅", "吳明"); // print 彭大牆
func("郭靜雅", "王立強", "郭林靜宜", "郭立恆", "林花花"); // print 林花花
func("郭宣雅", "林靜宜", "郭宣恆", "林靜花"); // print 沒有
func("郭宣雅", "夏曼藍波安", "郭宣恆"); // print 夏曼藍波安
```

Task 4:

There is a number sequence: 0, 4, 8, 7, 11, 15, 14, 18, 22, 21, 25, …

Find out the nth term in this sequence.

Note: Never change existing code.

Python

```
def get_number(index):
    # your code here
get_number(1) # print 4
get_number(5) # print 15
get_number(10) # print 25
get_number(30) # print 70
```

JavaScript

```
function getNumber(index){
    // your code here
}
getNumber(1); // print 4
getNumber(5); // print 15
getNumber(10); // print 25
getNumber(30); // print 70
```

Task 5: (Optional)

Given available spaces for each car of a train, status bitmap, and number of incoming passengers, writing a procedure to find out the index of the most fitted car to serve passengers. Print -1 if there is no car which can serve incoming passengers.

- Available Spaces: list/array containing number of available seats for each car.
- Status Bitmap: list/array containing only 0 or 1. 1 means the corresponding car can serve passengers for now.
- Passenger Number: number of incoming passengers.

We can assume all incoming passengers should be served in the same car.

Note: Never change existing code.

Python

```
def find(spaces, stat, n):
    # your code here
find([3, 1, 5, 4, 3, 2], [0, 1, 0, 1, 1, 1], 2) # print 5
find([1, 0, 5, 1, 3], [0, 1, 0, 1, 1], 4) # print -1
find([4, 6, 5, 8], [0, 1, 1, 1], 4) # print 2
```

JavaScript

```
function find(spaces, stat, n){
    // your code here
}
find([3, 1, 5, 4, 3, 2], [0, 1, 0, 1, 1, 1], 2); // print 5
find([1, 0, 5, 1, 3], [0, 1, 0, 1, 1], 4); // print -1
find([4, 6, 5, 8], [0, 1, 1, 1], 4); // print 2
```