1) *3 points*

Consider the below function.

```
def func(nums:list):
    return sum(num for num in nums if len(str(num))<2)
```

Select all the data processing pattern(s) found in the given function.

- [x] Aggregation
- [x] Filtering
- [ ] Mapping
- [ ] None of the above

2) *3 points*

Consider the below function.

```
def func(nums:list):
    return sum(num*2 for num in nums)
```

Select all the data processing pattern(s) found in the given function.

- [x] Aggregation
- [ ] Filtering
- [x] Mapping
- [ ] None of the above

3) *3 points*

Consider the below function.

```
def func(nums:list):
```

```
    return "".join((str(num) for num in nums))
```

Select all the data processing pattern(s) found in the given function.

- [x] Aggregation
- [ ] Filtering
- [x] Mapping
- [ ] None of the above

4) ***3 points***

Consider the below function.

```
def func(sentence:str):
    return next((word for word in sentence.split() if 'a' in word), None)
```

Select all the data processing pattern(s) found in the given function.

- [ ] Aggregation
- [x] Filtering
- [ ] Mapping
- [ ] None of the above

5) ***3 points***

Consider the below function.

```
def func(sentence:str):
    return [word*2 for word in words if 'a' in word]
```

Select all the data processing pattern(s) found in the given function.

- [ ] Aggregation
- [x] Filtering
- [x] Mapping
- [ ] None of the above

6) ***3 points***

Consider the below function.

```
def func(sentence:str):
    return [
        chr(ord(char)+1) if char in 'aeiou' else chr(ord(char)-1)
        for char in sentence
    ]
```

Select all the data processing pattern(s) found in the given function.

- [ ] Aggregation
- [ ] Filtering
- [x] Mapping
- [ ] None of the above

7) ***3 points***

Consider the below function.

```
def func(sentence:str):
    return ''.join([
        chr(ord(char)+1) if char in 'aeiou' else chr(ord(char)-1)
        for char in sentence
    ])
```

Select all the data processing pattern(s) found in the given function.

- [x] Aggregation
- [ ] Filtering
- [x] Mapping
- [ ] None of the above

8) ***3 points***

Consider the below function.

```
def func(sentence:str):
    return ''.join([
        chr(ord(char)+1)
        for char in sentence
        if char in 'aeiou'
    ])
```

Select all the data processing pattern(s) found in the given function.

- [ ] Aggregation
- [x] Filtering
- [x] Mapping
- [ ] None of the above

9) ***3 points***

Consider the below function.

```
def func(nums:list):
  return [num*2 for num in nums]
```

Select all the data processing pattern(s) found in the given function.

- [ ] Aggregation

- [ ] Filtering
- [x] Mapping
- [ ] None of the above

10) ***3 points***

Consider the below function.

```
def func(n:int):
    return [
        num*2 if len(str(num))<2 else num*3
        for num in range(n)
    ]
```

Select all the data processing pattern(s) found in the given function.

- [ ] Aggregation
- [ ] Filtering
- [x] Mapping
- [ ] None of the above

11) ***3 points***

Consider the below function.

```
def func(n:int):
    return [num for num in range(n) if len(str(num))<2]
```

Select all the data processing pattern(s) found in the given function.

- [ ] Aggregation
- [x] Filtering

- [ ] Mapping
- [ ] None of the above

12) ***3 points***

Consider the below function.

```
def func(n:int):
    return next((num for num in range(n) if len(str(num))<2), None)
```

Select all the data processing pattern(s) found in the given function.

- [ ] Aggregation
- [x] Filtering
- [ ] Mapping
- [ ] None of the above

13) ***3 points***

Consider the below function.

```
def func(nums:list):
    return min(num)
```

Select all the data processing pattern(s) found in the given function.

- [x] Aggregation
- [ ] Filtering
- [ ] Mapping
- [ ] None of the above

14) ***3 points***

Consider the below function.

```
def func(nums:list):
    return min(len(str(num)) for num in nums)
```

Select all the data processing pattern(s) found in the given function.

- [x] Aggregation
- [ ] Filtering
- [x] Mapping
- [ ] None of the above

15) ***3 points***

Consider the below function.

```
def func(nums:list):
    return min(nums, key= lambda x: len(str(abs(x)))
```

Select all the data processing pattern(s) found in the given function.

- [x] Aggregation
- [ ] Filtering
- [x] Mapping
- [ ] None of the above