Problems – Set 3

Q1) Create a function that takes a dictionary of objects like `{ "name": "John", "notes": [3, 5, 4] }` and returns a dictionary of objects like `{ "name": "John", "top_note": 5 }`.

Examples

```
top_note({ "name": "John", "notes": [3, 5, 4] }) → { "name": "John",
"top_note": 5 }

top_note({ "name": "Max", "notes": [1, 4, 6] }) → { "name": "Max",
"top_note": 6 }

top_note({ "name": "Zygmund", "notes": [1, 2, 3] }) → { "name":
"Zygmund", "top_note": 3 }
```

Q2) Hamming distance is the number of characters that differ between two strings.
To illustrate:

```
String1: "abcbba"
String2: "abcbda"

Hamming Distance: 1 - "b" vs. "d" is the only difference.
```

Create a function that computes the hamming distance between two strings.

Examples

```
hamming_distance("abcde", "bcdef") → 5

hamming_distance("abcde", "abcde") → 0

hamming_distance("strong", "strung") → 1
```

Notes

Both strings will have the same length.

Q3) An isogram is a word that has no duplicate letters. Create a function that takes a string and returns either `True` or `False` depending on whether or not it's an "isogram".

Examples

```
is_isogram("Algorism") → True

is_isogram("PasSword") → False
# Not case sensitive.

is_isogram("Consecutive") → False
```

Notes

- Ignore letter case (should not be case sensitive).
- All test cases contain valid one word strings.

Q4) Create a function that takes a dictionary of student names and returns a list of student names in alphabetical order.

Examples

```
get_student_names({
  "Student 1" : "Steve",
  "Student 2" : "Becky",
  "Student 3" : "John"
}) → ["Becky", "John", "Steve"]
```

Notes

- Don't forget to `return` your result.

Q5) Create a function that returns the mean of all digits.

Examples

```
mean(42) → 3

mean(12345) → 3

mean(666) → 6
```

Notes

- The mean of all digits is the sum of digits / how many digits there are (e.g. mean of digits in 512 is `(5+1+2)/3(number of digits) = 8/3=2`).
- The mean will always be an integer.

Q6) You are given a list of `dates` in the format `Dec 11` and a `month` in the format `Dec` as arguments. Each date represents a video that was uploaded on that day. Return the number of uploads for a given month.

Examples

```
upload_count(["Sept 22", "Sept 21", "Oct 15"], "Sept") → 2

upload_count(["Sept 22", "Sept 21", "Oct 15"], "Oct") → 1
```

Notes

If you only pay attention to the month and ignore the day, the challenge will become easier.

Q7) Create a function which adds spaces before every capital in a word. Uncapitalize the whole string afterwards.

Examples

```
cap_space("helloWorld") → "hello world"

cap_space("iLoveMyTeapot") → "i love my teapot"

cap_space("stayIndoors") → "stay indoors"
```

Notes

The first letter will stay uncapitalized.

Q8) Create a function that, given a number, returns the corresponding value of that index in the Fibonacci series.

The Fibonacci Sequence is the series of numbers:

```
1, 1, 2, 3, 5, 8, 13, 21, 34, ...
```

The next number is found by adding the two numbers before it:

- •The 2 is found by adding the two numbers before it (1+1).
- •The 3 is found by adding the two numbers before it (1+2).
- •The 5 is (2+3), and so on!

Examples

```
fibonacci(3) → 3

fibonacci(7) → 21

fibonacci(12) → 233
```

Notes

The first number in the sequence starts at 1 (not 0).

Q9) Given a list and chunk size "n", create a function such that it divides the list into many sublists where each sublist is of length size "n".

Examples

```
chunk([1, 2, 3, 4], 2) → [
  [1, 2], [3, 4]
]

chunk([1, 2, 3, 4, 5, 6, 7], 3) → [
  [1, 2, 3], [4, 5, 6], [7]
]

chunk([1, 2, 3, 4 ,5], 10) → [
  [1, 2, 3, 4, 5]
]
```

Notes

Remember that number of sublists may not be equal to chunk size.

Q10) You call your spouse to inform his/her most precious item is gone! Given a dictionary of stolen items, return the most expensive item on the items.

Examples

```
most_expensive_item({
  "piano": 2000,
}) → "piano"

most_expensive_item({
  "tv": 30,
  "skate": 20,
}) → "tv"

most_expensive_item({
  "tv": 30,
  "skate": 20,
  "stereo": 50,
}) → "stereo"
```

Notes

- There will only be one most valuable item (no ties).
- The dictionary will always contain at least one item (no empty dictionary).