

2. [8 marks] Programming.

(a) [3 marks]

Given the following (incomplete) function definition, complete it by (1) adding *one* doctest example, and (2) implementing the function body. You may **not** define any additional functions in your solution.

```
def compare_string_lengths(s1: str, s2: str, diff: int) -> bool:
    """Return whether <s1> has at least <diff> characters MORE than <s2>.
```

(Put another way, return whether <s1> is longer than <s2> by at least <diff> characters.)

Preconditions:

- diff >= 1

TODO: WRITE YOUR DOCTEST EXAMPLE HERE

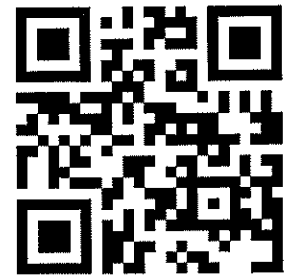
```
>>> compare_string_lengths('SHIVESH', 'TOM', 2)
True
```

"""

TODO: WRITE YOUR FUNCTION BODY HERE

```
difference = len(s1) - len(s2)
```

```
return difference >= diff
```



(b) [5 marks]

Given the following (incomplete) function definition, complete it by (1) adding *one* doctest example, and (2) implementing the function body. You **must** call `compare_string_lengths` from Part (a) when implementing this function. You may **not** define any additional functions in your solution.

```
def get_max_string_length(strings: list[str], s: str) -> int:
    """Return the length of the longest string in strings that is
    at least THREE (3) characters longer than s.
```

If there is no string in strings that meets the specifications, return -1 instead.

```
>>> get_max_string_length([], 'hello')
-1
```

TODO: WRITE YOUR DOCTEST EXAMPLE HERE. YOUR EXAMPLE MUST
SHOW A CALL TO THIS FUNCTION THAT RETURNS AN INT OTHER THAN -1.

```
>>> get_max_string_length(['TORONTO', 'CALIFORNIA'], 'LONDON')
10
```

```
"""
```

```
# TODO: WRITE YOUR FUNCTION BODY HERE
```

```
lengths = [len(x) for x in strings if compare_string_lengths(x, s, 3)]
```

```
if lengths == []:
    return -1
```

```
else:
    return max(lengths)
```