1. >>> help(str.replace)
Help on method_descriptor:
replace(...) -> str
    S.replace(old, new[, count]) -> str

Return a copy of S with all occurrences of substring
old replaced by new. If the optional argument count is
given, only the first count occurrences are replaced.

What is the output of this code?

s = 'Mississippi'
t = len(s.replace('ss', 'a'))
print(t)

A 11
B ss
C 9
D Miaaiaippi

2. >>> help(str.replace)
Help on method_descriptor:
replace(...) -> str
    S.replace(old, new[, count]) -> str

Return a copy of S with all occurrences of substring
old replaced by new. If the optional argument count is
given, only the first count occurrences are replaced.

What is the output of this code?

s = 'Mississippi'
t = len(s.replace('ss', 'a'))
print(t)

A 11
B ss
C 9
D Miaaiaippi

Recall that len returns the length of its argument, so t must
be an integer. Here all copies of "ss" "Mississippi" are
replaced with "a" to create "Miaaiaippi", which has length 9.
3. What is the value of `t` after this code executes?

```python
s = 'Mississippi'
t = s.upper().replace('ss', 'a')
```

- A. Mississippi
- B. Miaiaippi
- C. MISSISSIPPI
- D. MIAIAIPPI

4. What is the value of `t` after this code executes?

```python
s = 'Mississippi'
t = s.upper().replace('ss', 'a')
```

- A. Mississippi
- B. Miaiaippi
- C. MISSISSIPPI
- D. MIAIAIPPI

The upper method creates a new string "MISSISSIPPI" (all caps). When we invoke replace on that string (returned by upper), there are no instances of lower case "ss" (Python distinguishes between upper and lower case letters), and so no replacements will be performed. Thus t is "MISSISSIPPI"