NOTE: SCENARIO IS WHAT THE PUZZLE TAKER SEES.

SCENARIO:

Imagine you are developing a web framework in Python. One of the essential features of every web framework is HTTP file upload. Usually a web framework handles the entire process of file transfer. It receives the file/data over an HTTP connection, stores it in a temporary file, and at the end returns a file object (or the name of the temporary file) to the web application to do whatever it needs to do with the file/data. The following is a simple implementation of storing file/data into a temporary file. Consider the snippet of code below and answer the following questions, assuming that the code has all required permissions to execute.

```
01   import os
02   BUFFER = 4096
03
04   def store(stream):
05     path = os.tmpnam()      # get a unique temporary file path
06
07     file = open(path, 'wb') # open the temporary file
08     try:
09       bytes = stream.read(BUFFER)
10       while bytes:
111        file.write(bytes)
12         bytes = stream.read(BUFFER)
13     finally:
14       file.close()
15
16     return path            # return the path to the caller
```

Questions:
1. What will the `store` function do when executed?

2. Which of the following is correct if the framework is on production serving thousands of concurrent users?
a. The framework works without any issue if there's enough space for storage.
b. The framework fails because of too many IO operations.
c. The framework fails because of memory leakage in the implementation.
d. The framework might result to inconsistent data.
e. None of the above

[Other statistical questions will be imported here while creating survey.]

NOTE: ANSWER IS TO BE SHOWN TO THE PUZZLE TAKER AT THE END OF THE

SESSION.

ANSWER:
1. The function gets the data, generates a unique temporary file path, stores the data into the temporary file, and returns the path of the temporary file.

2. d
The `tmpnam` function generates a unique temporary file path. However, the maximum number of unique paths that the `tmpnam` function will generate before reusing names is determined by `TMP_MAX`. Depending on the circumstances, after thousands of calls, the generated file path might not be unique anymore, meaning the `store` function stores data to potentially existing files.

NOTE: THE REST OF THE DOCUMENT CONTAINS EXTRA INFORMATION FOR THE PROJECT RESEARCHERS. IT IS NOT TO BE SHOWN TO THE PUZZLE TAKERS.

TAGS:
python, io-operation, file-operation

NAME:
os.tmpnam()

CATEGORIES:
Blindspot - YES
Type - overflow
Number of distinct functions - 6
Number of total functions - 5
Blindspot function - `tmpnam()`
Function call omitted - NO
Blindspot type - Function misuse
Number of parameters in the blindspot function - 0 parameter
Cyclomatic complexity - 2

DESCRIPTION:
The method tmpnam returns a unique file path that is reasonable for creating a temporary file.

BLINDSPOT:
The blindspot of the tmpnam function is the maximum number of unique temporary file paths it can generate. This maximum number of unique paths that will generate before the `tmpnam` function reuses names is determined by `TMP_MAX`. Depending on the circumstances, after thousands of calls, the generated file path might not be unique anymore.

CORRECT USE EXAMPLE:
#N/A

MORE INFORMATION:
#N/A

REFERENCES:
1. https://docs.python.org/2/library/os.html#os.tmpnam
2. https://docs.python.org/2/library/os.html#os.TMP_MAX