OPERATING SYSTEM ASSIGNMENT #04

Particulars:

Name:	Asad Abbas
Reg No:	200901024
Date:	3-01-2023

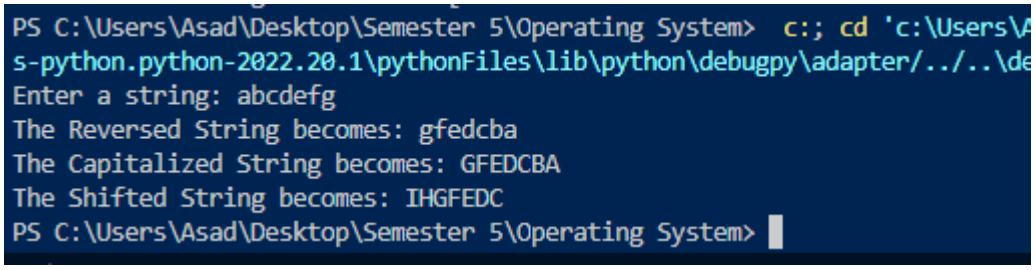
Code:

```
import threading

input_str = "" #store the input string


input_lock = threading.Lock()  #Lock to synchronize access to the input
string, preventing race conditions
```

```
def input_thread():  #Input thread function
  global input_str
  try:
    input_str = input("Enter a string: ")
  except EOFError:
    #Python stops the thread when there is an EOFError, allowing program to
exit when no further input is required
    return
  except Exception as e:
    #Print other exception (if any)
    print(e)


def reverse_thread(): #Thread to reverse the input string
  global input_str  #Globally defined above
  with input_lock:  #Acquire the input lock (used in all thread functions
just to be safe)
    input_str = input_str[::-1]  #Reverse string
    print("The Reversed String becomes:", input_str)

#Thread to capitalize the input string
def capital_thread():
  global input_str
  with input_lock:
    input_str = input_str.upper()  #Capitalizing input string using upper
function built in python
    print("The Capitalized String becomes:", input_str)


def shift_thread(): #Thread to shift the characters in the input string
  global input_str
  with input_lock:  #locking input here as well to prevent race condition
    # Shift the characters in the input string
    shifted_str = ""
    for ch in input_str:
      shifted_str += chr(ord(ch) + 2)  #Right shift of 2 characters
    input_str = shifted_str
    print("The Shifted String becomes:", input_str)

#Create threads
input_t = threading.Thread(target=input_thread)
reverse_t = threading.Thread(target=reverse_thread)
capital_t = threading.Thread(target=capital_thread)
shift_t = threading.Thread(target=shift_thread)

#Start the input thread
input_t.start()
```

```
# Wait for the input thread to finish
input_t.join()

#Start the other threads
reverse_t.start()
capital_t.start()
shift_t.start()

# Wait for the other threads to finish avoiding race condition
reverse_t.join()
capital_t.join()
shift_t.join()
```

Output:



Explanation:

We create four threads to perform different operations on a string entered as input. The input thread gets input from the user and stores it in the global **input_str** variable, while the reverse, capital, and shift threads each perform their respective operations on the input string and print the result.

Error Handling

The input thread uses a try-except block to handle any exceptions that might occur when getting input from the user, like EOFError. If an EOFError occurs, the input thread returns, allowing the program to exit gracefully. If any other exception occurs, it is printed to the console.

Prevention of Race Condition

After the input thread finishes, the main thread starts the other three threads using the **Thread.start()** method. The main thread then waits for all four threads to finish using the **Thread.join()** method, which blocks the main thread until the specified thread has completed. This helps to avoid race conditions or other synchronization issues between the threads.

To synchronize access to the **input_str** variable and prevent race conditions, the reverse, capital, and shift threads all use a **threading.Lock** object called **input_lock**. The lock is acquired at the beginning of each thread function using the **with** statement, and it is released when the **with** block ends. This ensures that only one thread can access the **input_str** variable at a time, which can help to prevent issues such as data corruption or inconsistent results.

Overall, these four threads concurrently perform different operations on a string entered by the user, using a lock to synchronize access to the input string and avoid race conditions. The main thread waits for all four threads to finish before exiting, ensuring that all the operations are completed before the program ends.

GitHub Link:

https://github.com/asadnaqviii/Semester-5--OS---CC