

**Manisa Celal Bayar University – Department of Computer Engineering**  
**CSE 3237 Parallel Programming – Midterm Exam**

Name and Surname	
Student Id	
Signature	

Question	1	2	3	4	Total
Score					

**Questions**

**Q1 (25 Points)** The Python program given below fetches the from a URL which returns a dictionary including the prime numbers below an integer **n**. Please fill the empty area with the definition of **scheduling** function/coroutine to send multiple requests to this URL asynchronously.

```
import asyncio
import aiohttp
```

```
async def get_primes_below(number: int, session: aiohttp.ClientSession) -> list:
    print(f"Getting primes below {number}")
    headers = {
        "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 12_6) "
        "AppleWebKit/537.36 (KHTML, like Gecko) "
        "Chrome/106.0.0.0 Safari/537.36"
    }
    url = f"https://www.canbula.com/prime/{number}"
    async with session.get(url, headers=headers, ssl=None) as response:
        primes = (await response.json())["primes"]
    print(f"Got primes below {number}: {primes}")
    return primes
```

```
if __name__ == "__main__":
    asyncio.run(scheduling([50, 10, 30, 40, 20]))
```

**Q2 (25 Points)** Write a Python **context manager** with the name `ThreadGenerator` which generates threads from a list of functions as given in the code below. `ThreadGenerator` must satisfy the following requirements:

- Threads must be started when the `with` statement is entered.
- The code inside the `with` block must be executed concurrently with the threads.
- Threads must be joined when the `with` statement is exited.

```
import threading
import time

class ThreadGenerator:
    def __init__(self, functions: list = []):
        self.functions: list = functions
        self.threads: list = []
```

```
def main():
    def f1():
        time.sleep(1)
        print("f1")

    def f2():
        time.sleep(5)
        print("f2")

    def f3():
        time.sleep(3)
        print("f3")

    with ThreadGenerator([f1, f2, f3]):
        print("Hello")

if __name__ == "__main__":
    main()
```

**Q3 (25 Points)** The `Wallet` class given in the code below will be used in many threads. Modify or rewrite this class to be sure that no **race condition** occurs.

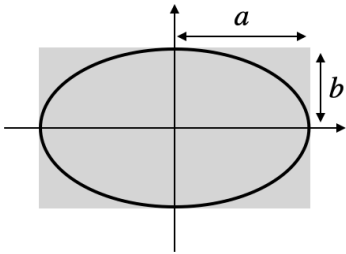
```
import threading

class Wallet:
    def __init__(self):
        self.balance = 0

    def deposit(self, amount):
        self.balance += amount

    def withdraw(self, amount):
        self.balance -= amount
```

**Q4 (25 Points)** Write a Python program to estimate the area of an ellipse using the Monte Carlo method. Please make your code should be using **multiple threads** but not suffering from **GIL**.



Points in the ellipse follows:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

Area of an ellipse is given as:

$$\pi \cdot a \cdot b$$

	QUESTIONS VS PCB MATRIX																											
	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Q1	✓	✓					✓	✓																				
Q2	✓	✓						✓																				
Q3	✓	✓					✓	✓																				
Q4	✓	✓																										