Set 5

1. Q1. Generate Python code for the Fibonacci sequence with Cursor AI. Refine your prompts to handle only positive integers and record the changes and outcomes.

(Each question carries 5 marks)

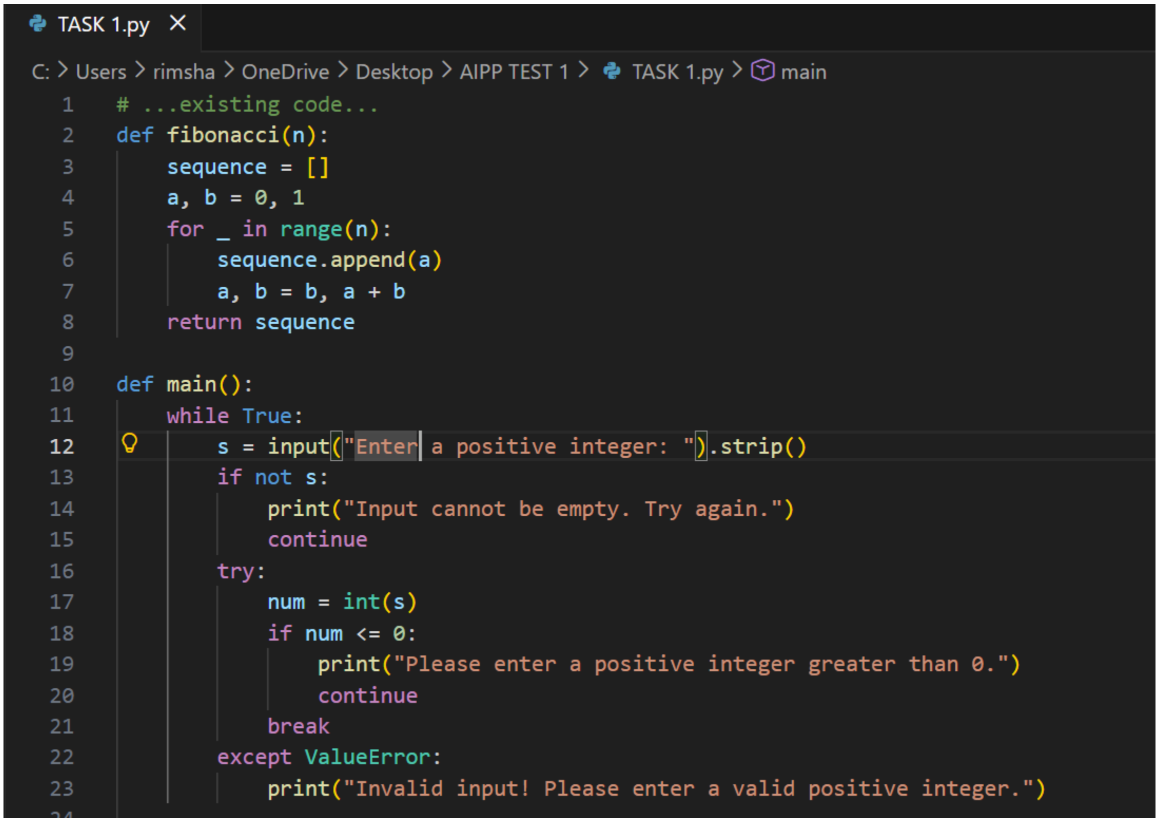
2. Q2. Write a Python queue class (`enqueue`, `dequeue`, `peek`). Use an AI assistant and test the class with multiple scenarios. Describe how AI-generated code manages edge cases and how you improved it.

(Each question carries 5 marks)

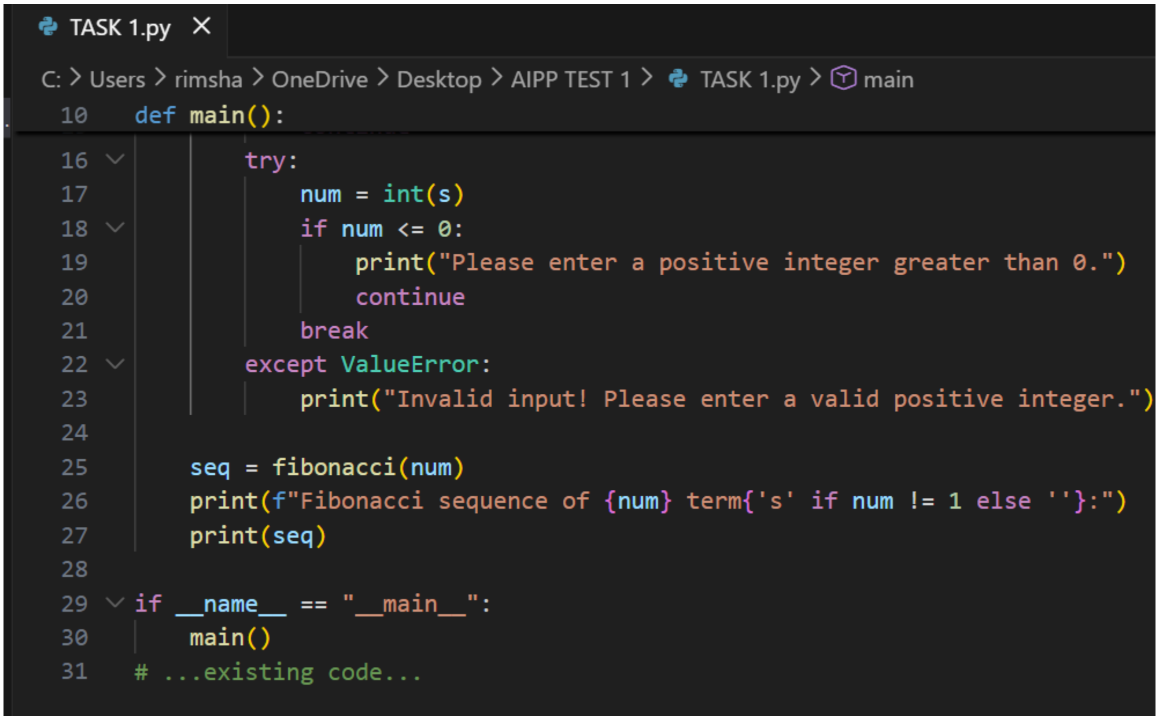
ANSWERS:

Q1. Generate Python code for the Fibonacci sequence with Cursor AI. Refine your prompts to handle only positive integers and record the changes and outcomes.

PROMPT: Write Python code that takes a positive integer as user input and prints the Fibonacci sequence with input validation. Include example output.

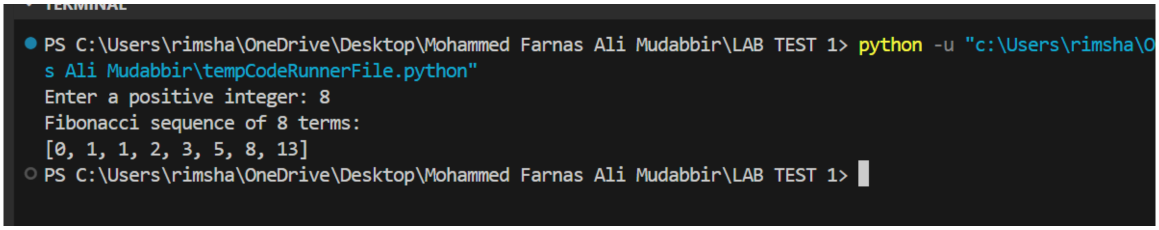

TASK 1.py
C: > Users > rimsha > OneDrive > Desktop > AIPP TEST 1 > TASK 1.py > main

```python
# ...existing code...
def fibonacci(n):
    sequence = []
    a, b = 0, 1
    for _ in range(n):
        sequence.append(a)
        a, b = b, a + b
    return sequence

def main():
    while True:
        s = input("Enter a positive integer: ").strip()
        if not s:
            print("Input cannot be empty. Try again.")
            continue
        try:
            num = int(s)
            if num <= 0:
                print("Please enter a positive integer greater than 0.")
                continue
            break
        except ValueError:
            print("Invalid input! Please enter a valid positive integer.")
```


TASK 1.py
C: > Users > rimsha > OneDrive > Desktop > AIPP TEST 1 > TASK 1.py > main

```python
def main():
        try:
            num = int(s)
            if num <= 0:
                print("Please enter a positive integer greater than 0.")
                continue
            break
        except ValueError:
            print("Invalid input! Please enter a valid positive integer.")

    seq = fibonacci(num)
    print(f"Fibonacci sequence of {num} term{'s' if num != 1 else ''}:")
    print(seq)

if __name__ == "__main__":
    main()
# ...existing code...
```

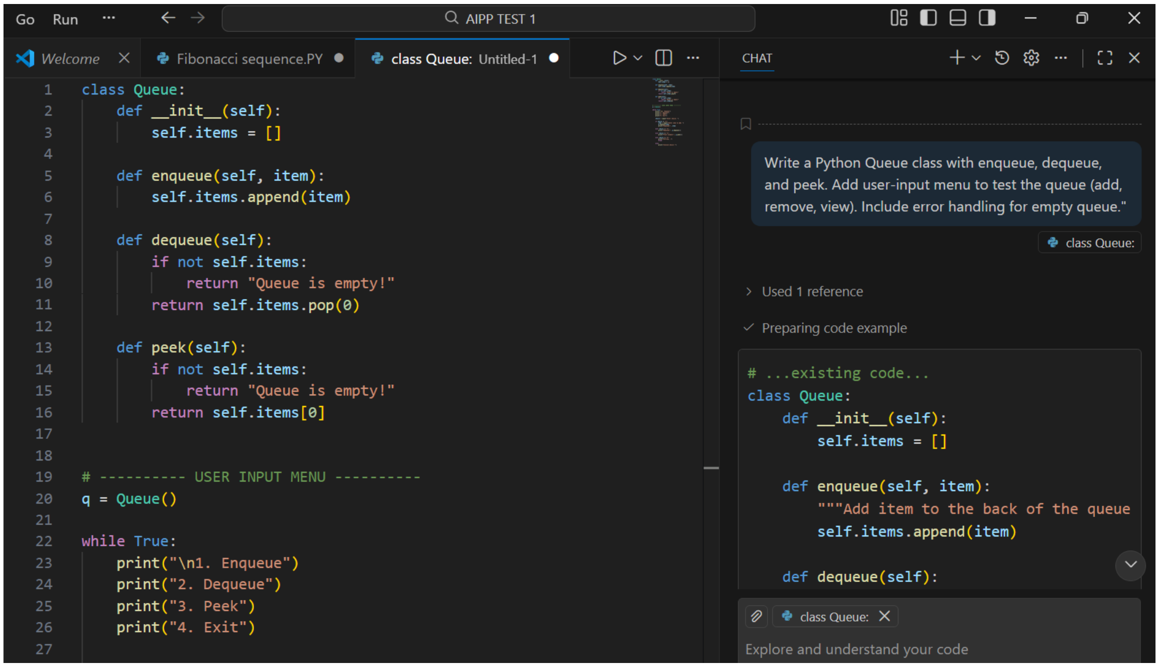
PRACTICAL OUTPUT:



```
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB TEST 1> python -u "c:\Users\rimsha\O
s Ali Mudabbir\tempCodeRunnerFile.python"
Enter a positive integer: 8
Fibonacci sequence of 8 terms:
[0, 1, 1, 2, 3, 5, 8, 13]
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB TEST 1> ▌
```

Q2. Write a Python queue class (`enqueue`, `dequeue`, `peek`). Use an AI assistant and test the class with multiple scenarios. Describe how AI-generated code manages edge cases and how you improved it.

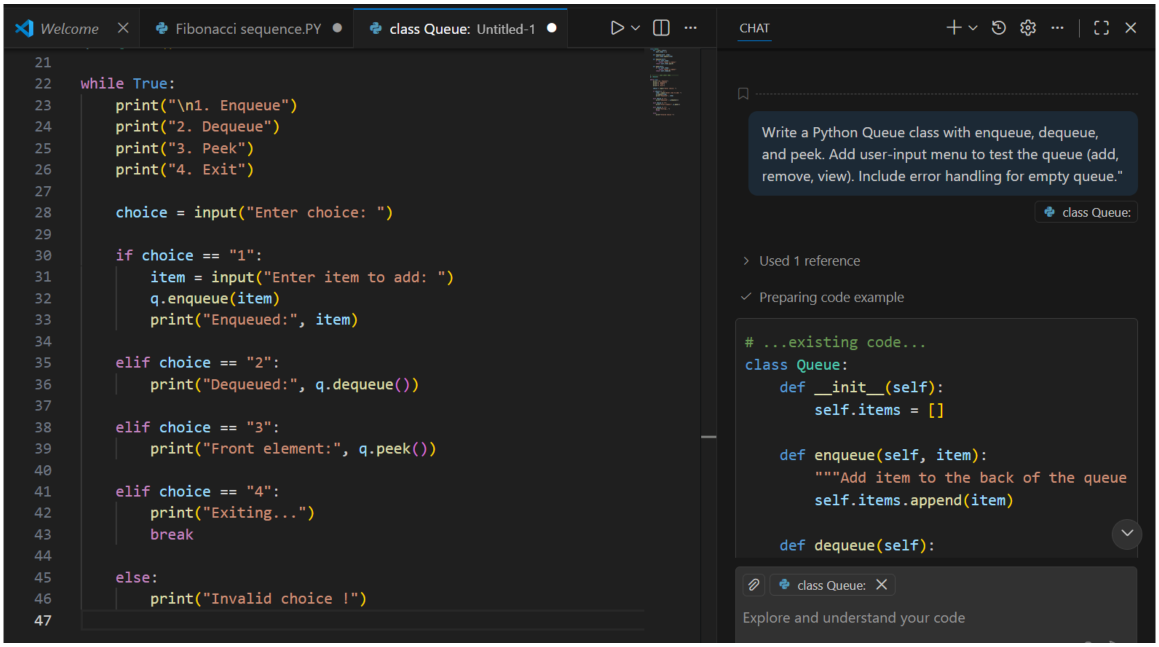
PROMPT: Write a Python Queue class with enqueue, dequeue, and peek. Add user-input menu to test the queue (add, remove, view). Include error handling for empty queue."



```python
class Queue:
    def __init__(self):
        self.items = []

    def enqueue(self, item):
        self.items.append(item)

    def dequeue(self):
        if not self.items:
            return "Queue is empty!"
        return self.items.pop(0)

    def peek(self):
        if not self.items:
            return "Queue is empty!"
        return self.items[0]


# ---------- USER INPUT MENU ----------
q = Queue()

while True:
    print("\n1. Enqueue")
    print("2. Dequeue")
    print("3. Peek")
    print("4. Exit")

```

Write a Python Queue class with enqueue, dequeue, and peek. Add user-input menu to test the queue (add, remove, view). Include error handling for empty queue."

```python
# ...existing code...
class Queue:
    def __init__(self):
        self.items = []

    def enqueue(self, item):
        """Add item to the back of the queue
        self.items.append(item)

    def dequeue(self):
```



```python

while True:
    print("\n1. Enqueue")
    print("2. Dequeue")
    print("3. Peek")
    print("4. Exit")

    choice = input("Enter choice: ")

    if choice == "1":
        item = input("Enter item to add: ")
        q.enqueue(item)
        print("Enqueued:", item)

    elif choice == "2":
        print("Dequeued:", q.dequeue())

    elif choice == "3":
        print("Front element:", q.peek())

    elif choice == "4":
        print("Exiting...")
        break

    else:
        print("Invalid choice !")

```

Write a Python Queue class with enqueue, dequeue, and peek. Add user-input menu to test the queue (add, remove, view). Include error handling for empty queue."

class Queue:

Used 1 reference

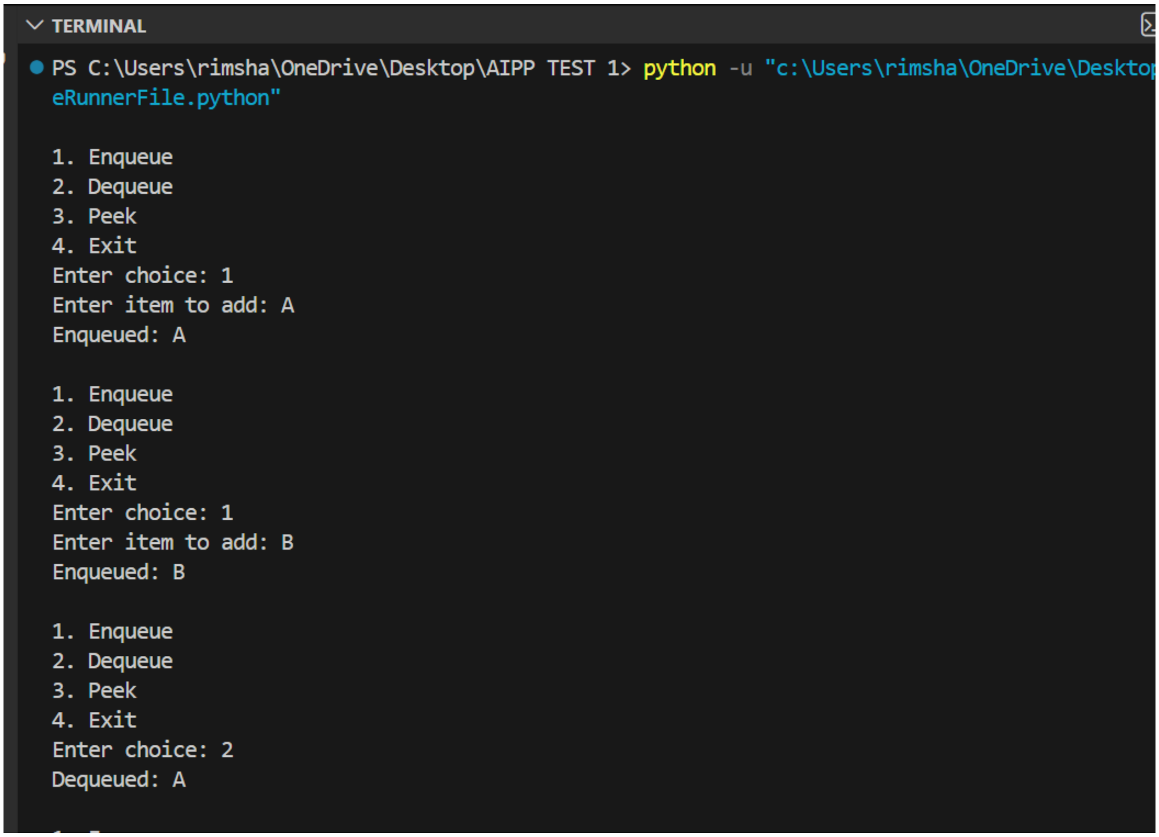
Preparing code example

```python
# ...existing code...
class Queue:
    def __init__(self):
        self.items = []

    def enqueue(self, item):
        """Add item to the back of the queue
        self.items.append(item)

    def dequeue(self):
```

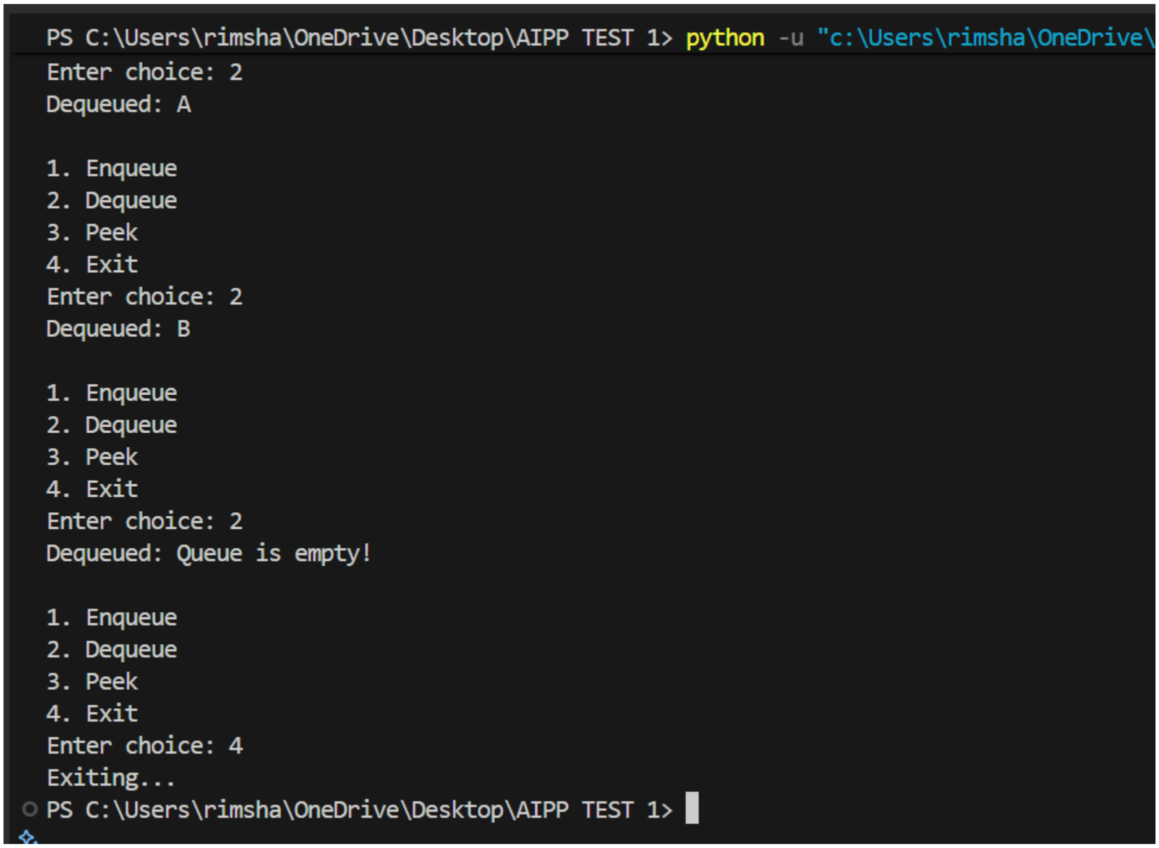
class Queue:

Explore and understand your code

PRACTICAL OUT:



```
PS C:\Users\rimsha\OneDrive\Desktop\AIPP TEST 1> python -u "c:\Users\rimsha\OneDrive\Deskto
eRunnerFile.python"

1. Enqueue
2. Dequeue
3. Peek
4. Exit
Enter choice: 1
Enter item to add: A
Enqueued: A

1. Enqueue
2. Dequeue
3. Peek
4. Exit
Enter choice: 1
Enter item to add: B
Enqueued: B

1. Enqueue
2. Dequeue
3. Peek
4. Exit
Enter choice: 2
Dequeued: A
```



```
PS C:\Users\rimsha\OneDrive\Desktop\AIPP TEST 1> python -u "c:\Users\rimsha\OneDrive\
Enter choice: 2
Dequeued: A

1. Enqueue
2. Dequeue
3. Peek
4. Exit
Enter choice: 2
Dequeued: B

1. Enqueue
2. Dequeue
3. Peek
4. Exit
Enter choice: 2
Dequeued: Queue is empty!

1. Enqueue
2. Dequeue
3. Peek
4. Exit
Enter choice: 4
Exiting...
PS C:\Users\rimsha\OneDrive\Desktop\AIPP TEST 1>
```