Sem-III **Harshvardhan Pandey**

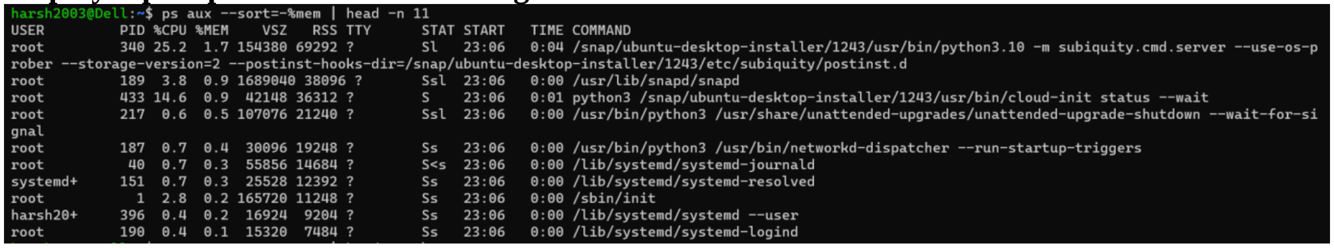
SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

A.Y.: 2023-24 **Class: S.Y.B.Tech** **Sub:** System Fundamentals

System Fundamentals Experiment List

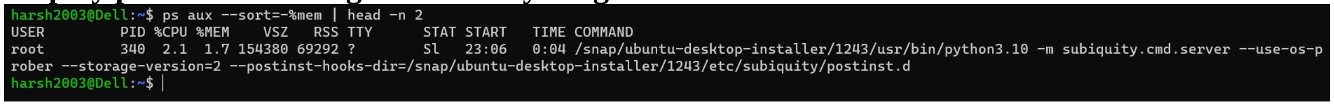
Explore the internal commands of Linux and Write shell scripts to do the following:

1. **Display top 10 processes in descending order**



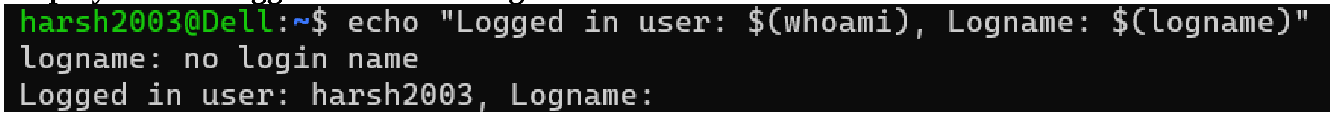
```
harsh2003@Dell:~$ ps aux --sort=-%mem | head -n 11
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         340 25.2  1.7 154380 69292 ?        Sl   23:06   0:04 /snap/ubuntu-desktop-installer/1243/usr/bin/python3.10 -m subiquity.cmd.server --use-os-p
rober --storage-version=2 --postinst-hooks-dir=/snap/ubuntu-desktop-installer/1243/etc/subiquity/postinst.d
root         189  3.8  0.9 1689040 38096 ?       Ssl  23:06   0:00 /usr/lib/snapd/snapd
root         433 14.6  0.9  42148 36312 ?        S    23:06   0:01 python3 /snap/ubuntu-desktop-installer/1243/usr/bin/cloud-init status --wait
root         217  0.6  0.5 107076 21240 ?        Ssl  23:06   0:00 /usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgrade-shutdown --wait-for-si
gnal
root         187  0.7  0.4  30096 19248 ?        Ss   23:06   0:00 /usr/bin/python3 /usr/bin/networkd-dispatcher --run-startup-triggers
root          40  0.7  0.3  55856 14684 ?        S<s  23:06   0:00 /lib/systemd/systemd-journald
systemd+     151  0.7  0.3  25528 12392 ?        Ss   23:06   0:00 /lib/systemd/systemd-resolved
root           1  2.8  0.2 165720 11248 ?        Ss   23:06   0:00 /sbin/init
harsh20+     396  0.4  0.2  16924  9204 ?        Ss   23:06   0:00 /lib/systemd/systemd --user
root         190  0.4  0.1  15320  7484 ?        Ss   23:06   0:00 /lib/systemd/systemd-logind
```

2. **Display processes with highest memory usage.**



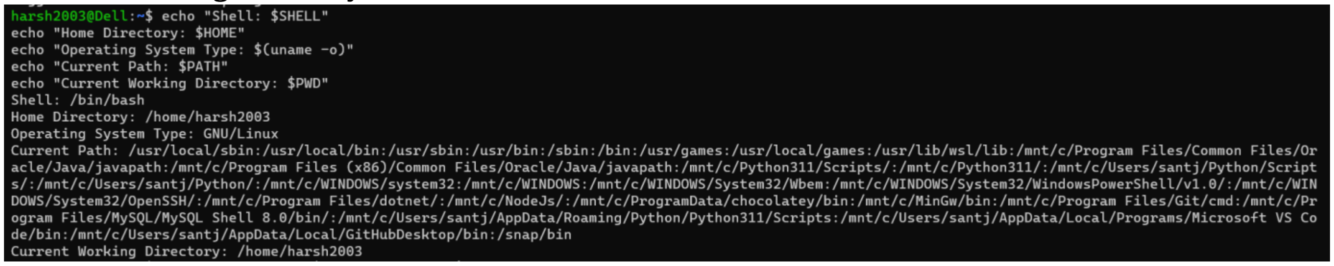
```
harsh2003@Dell:~$ ps aux --sort=-%mem | head -n 2
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         340  2.1  1.7 154380 69292 ?        Sl   23:06   0:04 /snap/ubuntu-desktop-installer/1243/usr/bin/python3.10 -m subiquity.cmd.server --use-os-p
rober --storage-version=2 --postinst-hooks-dir=/snap/ubuntu-desktop-installer/1243/etc/subiquity/postinst.d
harsh2003@Dell:~$ |
```

3. **Display current logged in user and logname.**



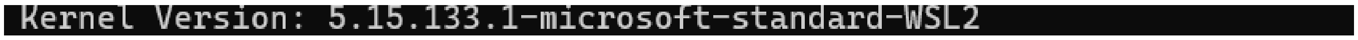
```
harsh2003@Dell:~$ echo "Logged in user: $(whoami), Logname: $(logname)"
logname: no login name
Logged in user: harsh2003, Logname:
```

4. **Display current shell, home directory, operating system type, current path setting, current working directory.**



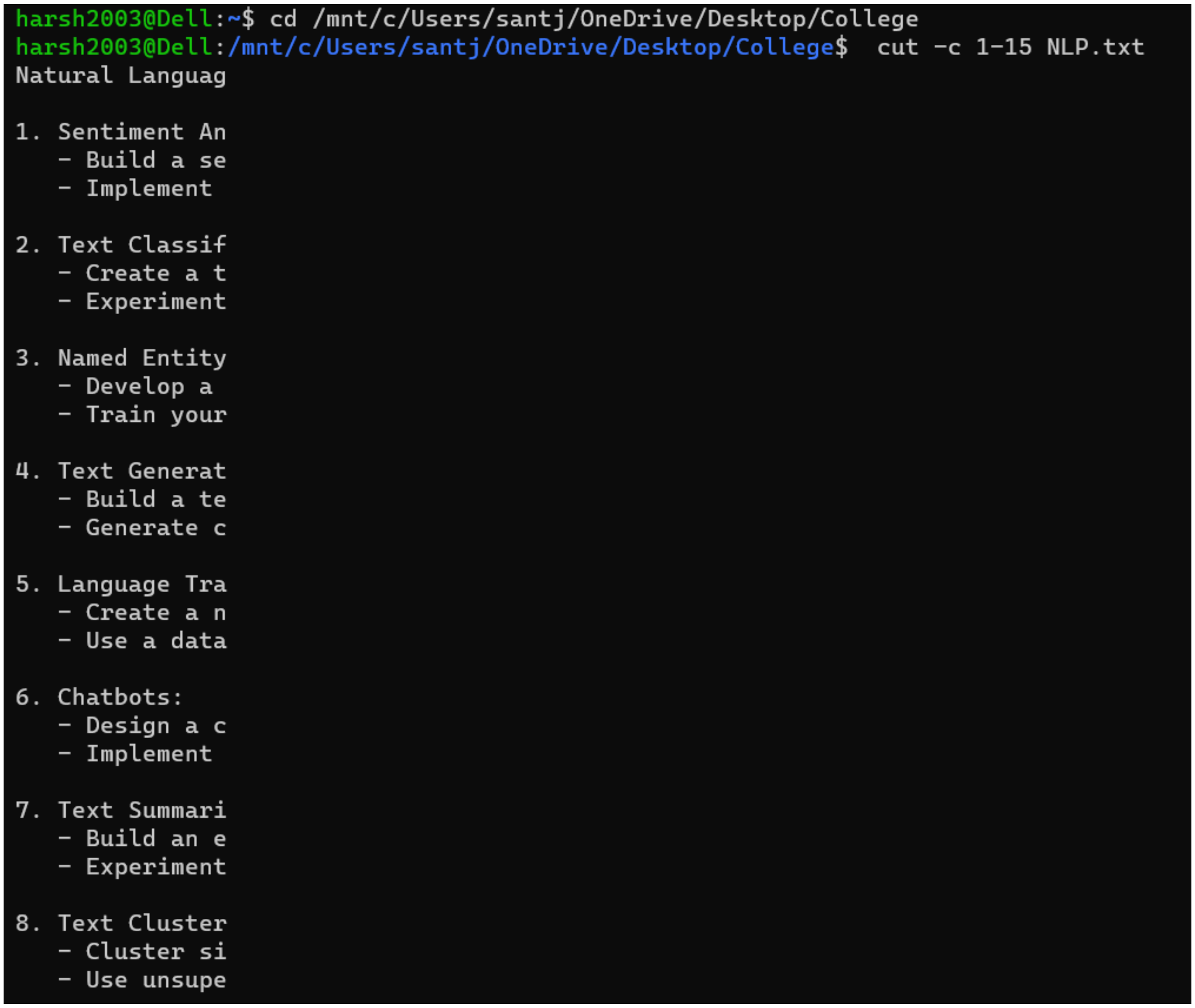
```
harsh2003@Dell:~$ echo "Shell: $SHELL"
echo "Home Directory: $HOME"
echo "Operating System Type: $(uname -o)"
echo "Current Path: $PATH"
echo "Current Working Directory: $PWD"
Shell: /bin/bash
Home Directory: /home/harsh2003
Operating System Type: GNU/Linux
Current Path: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/lib/wsl/lib:/mnt/c/Program Files/Common Files/Or
acle/Java/javapath:/mnt/c/Program Files (x86)/Common Files/Oracle/Java/javapath:/mnt/c/Python311/Scripts/:/mnt/c/Python311/:/mnt/c/Users/santj/Python/Script
s/:/mnt/c/Users/santj/Python/:/mnt/c/WINDOWS/system32:/mnt/c/WINDOWS:/mnt/c/WINDOWS/System32/Wbem:/mnt/c/WINDOWS/System32/WindowsPowerShell/v1.0/:/mnt/c/WIN
DOWS/System32/OpenSSH/:/mnt/c/Program Files/dotnet/:/mnt/c/NodeJs/:/mnt/c/ProgramData/chocolatey/bin:/mnt/c/MinGw/bin:/mnt/c/Program Files/Git/cmd:/mnt/c/Pr
ogram Files/MySQL/MySQL Shell 8.0/bin/:/mnt/c/Users/santj/AppData/Roaming/Python/Python311/Scripts:/mnt/c/Users/santj/AppData/Local/Programs/Microsoft VS Co
de/bin:/mnt/c/Users/santj/AppData/Local/GitHubDesktop/bin:/snap/bin
Current Working Directory: /home/harsh2003
```

5. **Display OS version, release number, kernel version.**



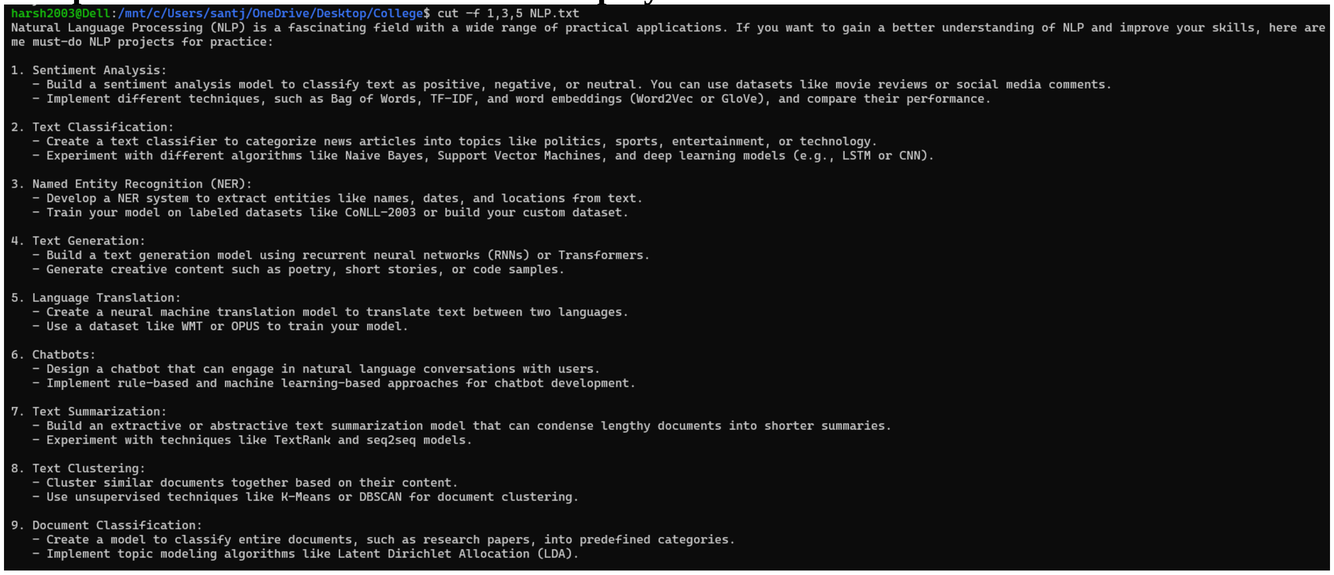
```
Kernel Version: 5.15.133.1-microsoft-standard-WSL2
```

6. **Write a command to display the first 15 columns from each line in the file**



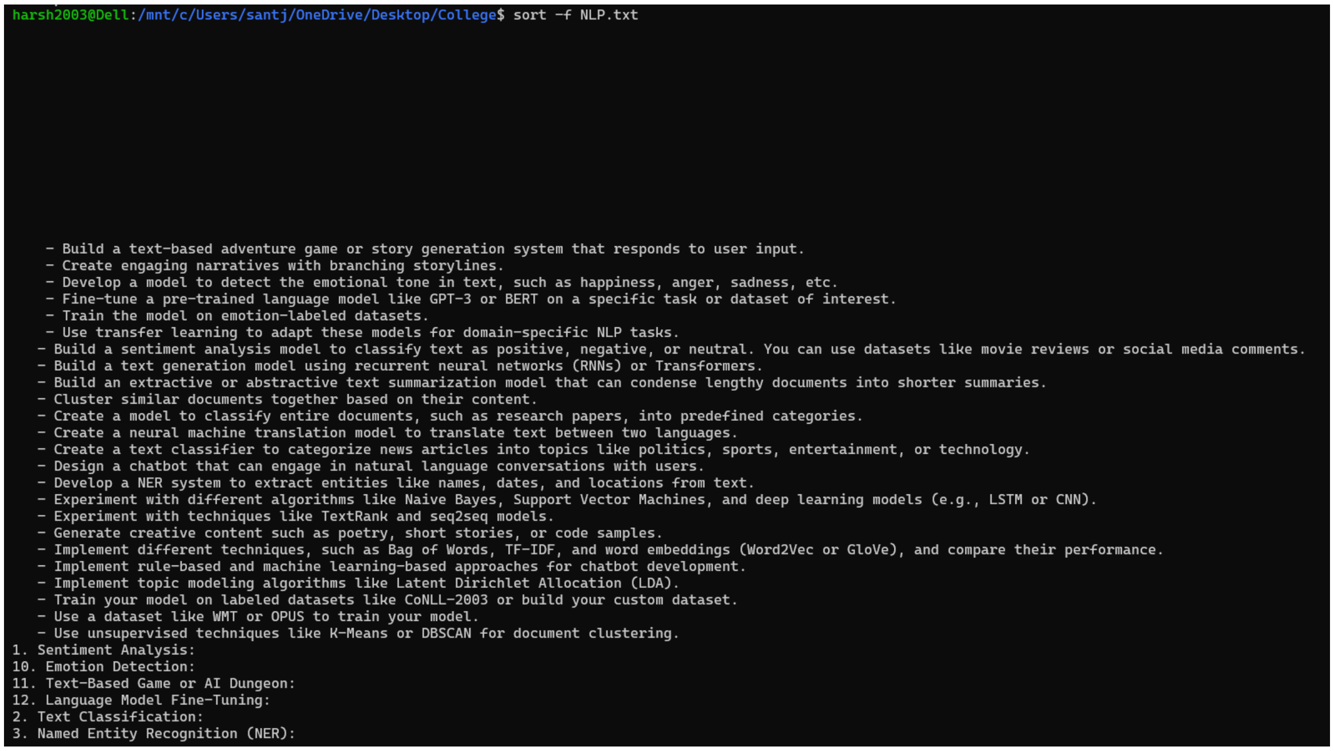
```
harsh2003@Dell:~$ cd /mnt/c/Users/santj/OneDrive/Desktop/College
harsh2003@Dell:/mnt/c/Users/santj/OneDrive/Desktop/College$  cut -c 1-15 NLP.txt
Natural Languag

1. Sentiment An
   - Build a se
   - Implement

2. Text Classif
   - Create a t
   - Experiment

3. Named Entity
   - Develop a
   - Train your

4. Text Generat
   - Build a te
   - Generate c

5. Language Tra
   - Create a n
   - Use a data

6. Chatbots:
   - Design a c
   - Implement

7. Text Summari
   - Build an e
   - Experiment

8. Text Cluster
   - Cluster si
   - Use unsupe
```

7. **cut specified columns from a file and display them**



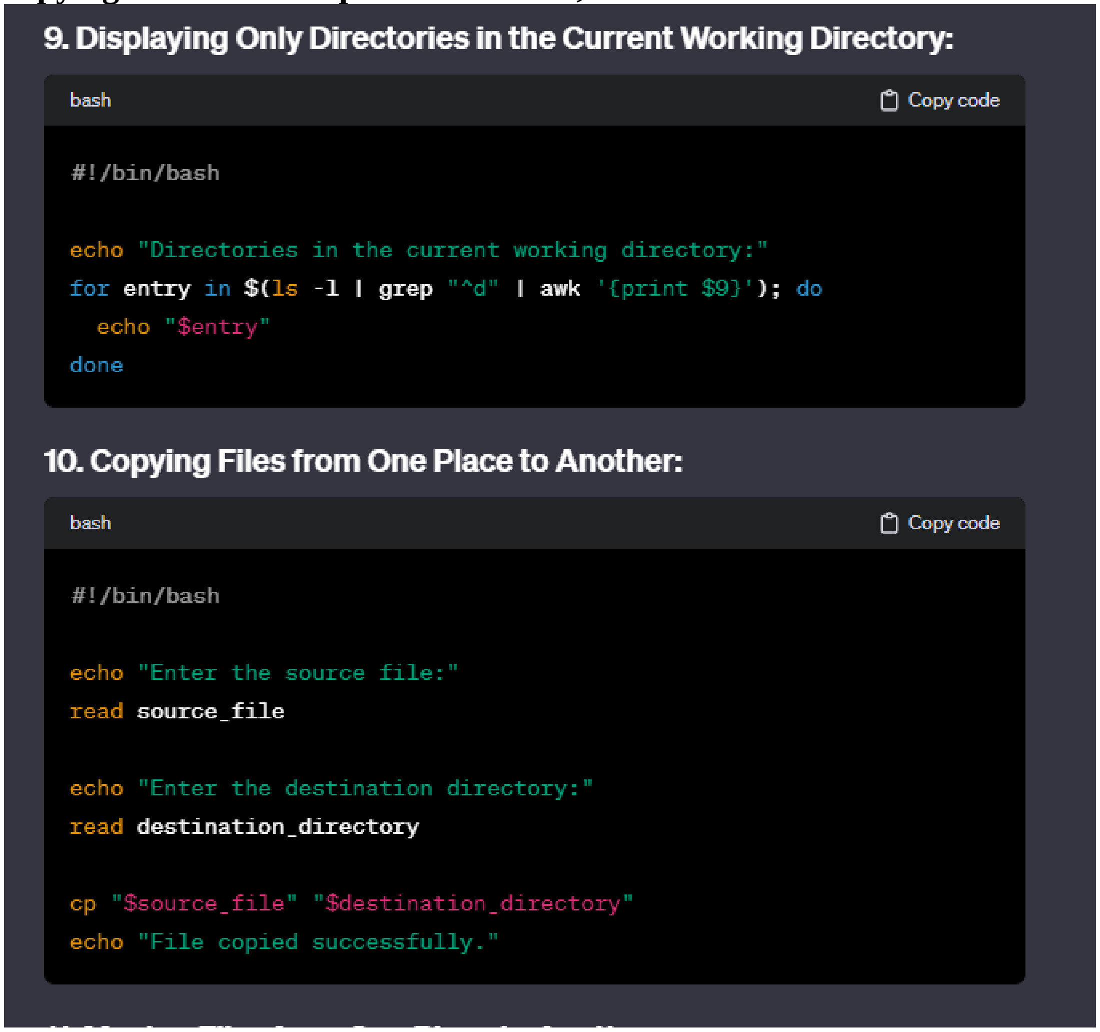
```
harsh2003@Dell:/mnt/c/Users/santj/OneDrive/Desktop/College$ cut -f 1,3,5 NLP.txt
Natural Language Processing (NLP) is a fascinating field with a wide range of practical applications. If you want to gain a better understanding of NLP and improve your skills, here are
me must-do NLP projects for practice:

1. Sentiment Analysis:
   - Build a sentiment analysis model to classify text as positive, negative, or neutral. You can use datasets like movie reviews or social media comments.
   - Implement different techniques, such as Bag of Words, TF-IDF, and word embeddings (Word2Vec or GloVe), and compare their performance.

2. Text Classification:
   - Create a text classifier to categorize news articles into topics like politics, sports, entertainment, or technology.
   - Experiment with different algorithms like Naive Bayes, Support Vector Machines, and deep learning models (e.g., LSTM or CNN).

3. Named Entity Recognition (NER):
   - Develop a NER system to extract entities like names, dates, and locations from text.
   - Train your model on labeled datasets like CoNLL-2003 or build your custom dataset.

4. Text Generation:
   - Build a text generation model using recurrent neural networks (RNNs) or Transformers.
   - Generate creative content such as poetry, short stories, or code samples.

5. Language Translation:
   - Create a neural machine translation model to translate text between two languages.
   - Use a dataset like WMT or OPUS to train your model.

6. Chatbots:
   - Design a chatbot that can engage in natural language conversations with users.
   - Implement rule-based and machine learning-based approaches for chatbot development.

7. Text Summarization:
   - Build an extractive or abstractive text summarization model that can condense lengthy documents into shorter summaries.
   - Experiment with techniques like TextRank and seq2seq models.

8. Text Clustering:
   - Cluster similar documents together based on their content.
   - Use unsupervised techniques like K-Means or DBSCAN for document clustering.

9. Document Classification:
   - Create a model to classify entire documents, such as research papers, into predefined categories.
   - Implement topic modeling algorithms like Latent Dirichlet Allocation (LDA).
```

8. **Sort given file ignoring upper and lower case**



```
harsh2003@Dell:/mnt/c/Users/santj/OneDrive/Desktop/College$ sort -f NLP.txt









    - Build a text-based adventure game or story generation system that responds to user input.
    - Create engaging narratives with branching storylines.
    - Develop a model to detect the emotional tone in text, such as happiness, anger, sadness, etc.
    - Fine-tune a pre-trained language model like GPT-3 or BERT on a specific task or dataset of interest.
    - Train the model on emotion-labeled datasets.
    - Use transfer learning to adapt these models for domain-specific NLP tasks.
   - Build a sentiment analysis model to classify text as positive, negative, or neutral. You can use datasets like movie reviews or social media comments.
   - Build a text generation model using recurrent neural networks (RNNs) or Transformers.
   - Build an extractive or abstractive text summarization model that can condense lengthy documents into shorter summaries.
   - Cluster similar documents together based on their content.
   - Create a model to classify entire documents, such as research papers, into predefined categories.
   - Create a neural machine translation model to translate text between two languages.
   - Create a text classifier to categorize news articles into topics like politics, sports, entertainment, or technology.
   - Design a chatbot that can engage in natural language conversations with users.
   - Develop a NER system to extract entities like names, dates, and locations from text.
   - Experiment with different algorithms like Naive Bayes, Support Vector Machines, and deep learning models (e.g., LSTM or CNN).
   - Experiment with techniques like TextRank and seq2seq models.
   - Generate creative content such as poetry, short stories, or code samples.
   - Implement different techniques, such as Bag of Words, TF-IDF, and word embeddings (Word2Vec or GloVe), and compare their performance.
   - Implement rule-based and machine learning-based approaches for chatbot development.
   - Implement topic modeling algorithms like Latent Dirichlet Allocation (LDA).
   - Train your model on labeled datasets like CoNLL-2003 or build your custom dataset.
   - Use a dataset like WMT or OPUS to train your model.
   - Use unsupervised techniques like K-Means or DBSCAN for document clustering.
1. Sentiment Analysis:
10. Emotion Detection:
11. Text-Based Game or AI Dungeon:
12. Language Model Fine-Tuning:
2. Text Classification:
3. Named Entity Recognition (NER):
```

9. **Displays only directories in current working directory.**
10. **copying files from one place to another,**



9. Displaying Only Directories in the Current Working Directory:

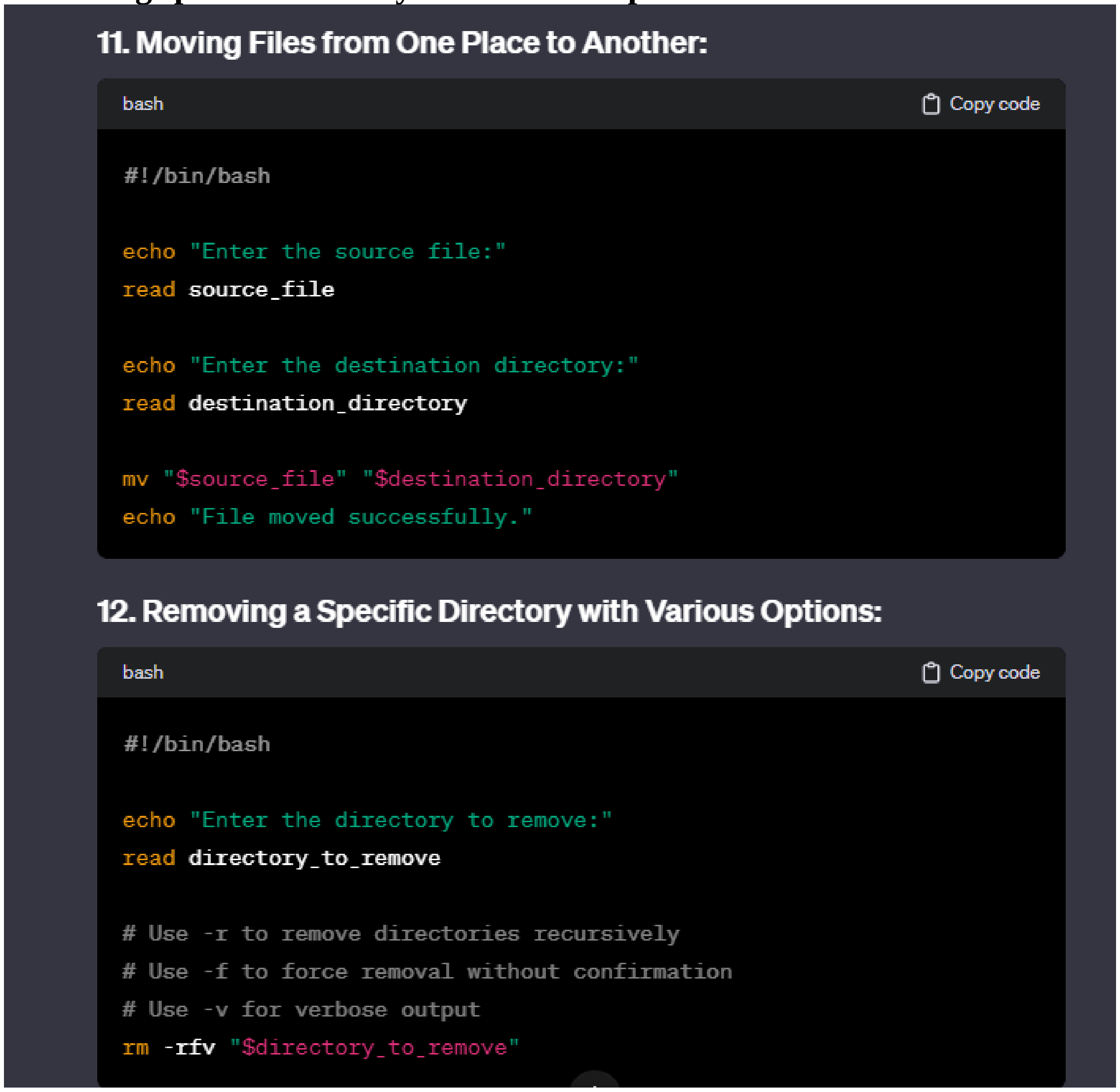
```bash
#!/bin/bash

echo "Directories in the current working directory:"
for entry in $(ls -l | grep "^d" | awk '{print $9}'); do
  echo "$entry"
done
```

10. Copying Files from One Place to Another:

```bash
#!/bin/bash

echo "Enter the source file:"
read source_file

echo "Enter the destination directory:"
read destination_directory

cp "$source_file" "$destination_directory"
echo "File copied successfully."
```

11. **moving files from one place to another.**

12. **Removing specific directory with various options**



11. Moving Files from One Place to Another:

bash Copy code

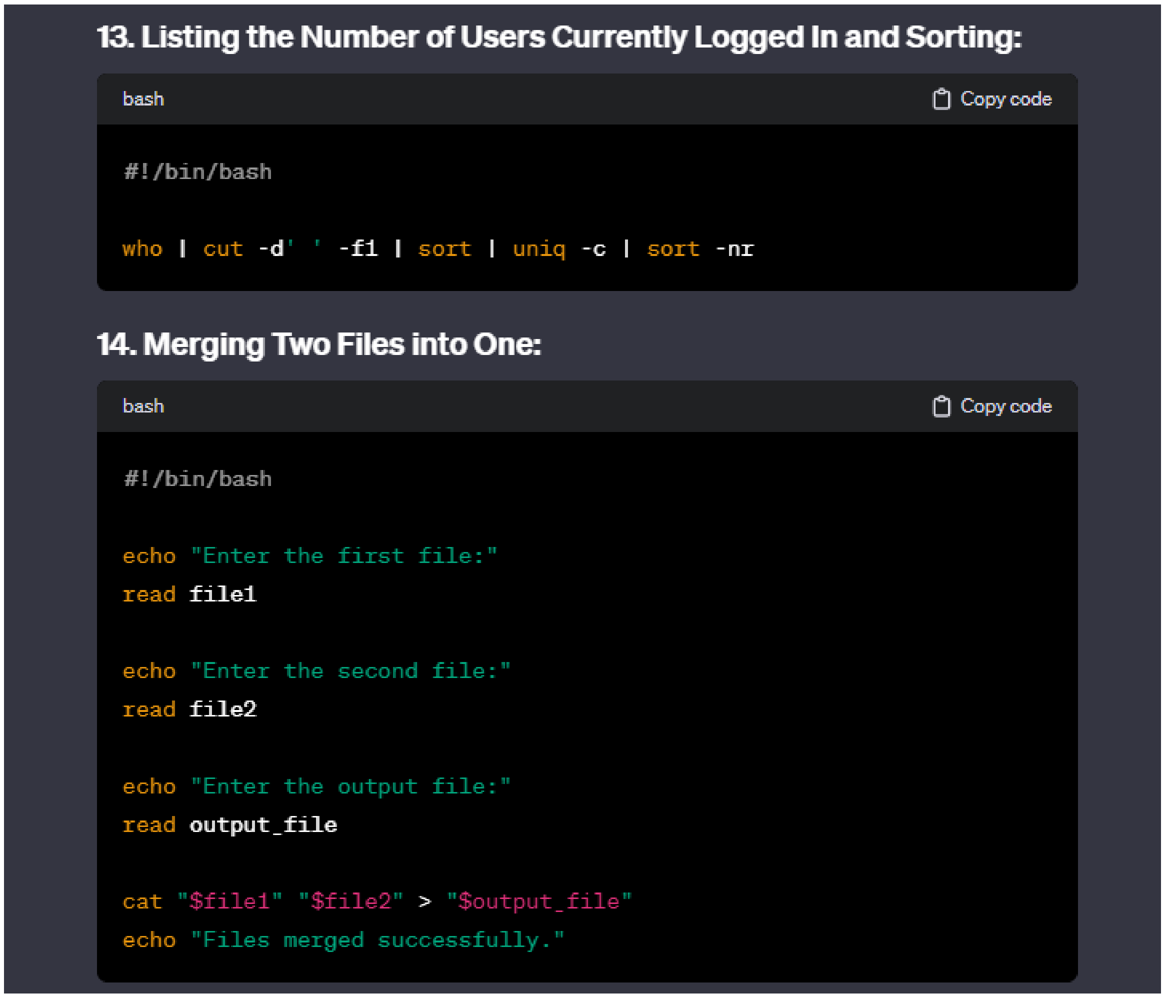
```bash
#!/bin/bash

echo "Enter the source file:"
read source_file

echo "Enter the destination directory:"
read destination_directory

mv "$source_file" "$destination_directory"
echo "File moved successfully."
```

12. Removing a Specific Directory with Various Options:

bash Copy code

```bash
#!/bin/bash

echo "Enter the directory to remove:"
read directory_to_remove

# Use -r to remove directories recursively
# Use -f to force removal without confirmation
# Use -v for verbose output
rm -rfv "$directory_to_remove"
```

13. **list the numbers of users currently login in the system and then sort it.**
14. **Merge two files into one file**



13. Listing the Number of Users Currently Logged In and Sorting:

bash Copy code

```bash
#!/bin/bash

who | cut -d' ' -f1 | sort | uniq -c | sort -nr
```

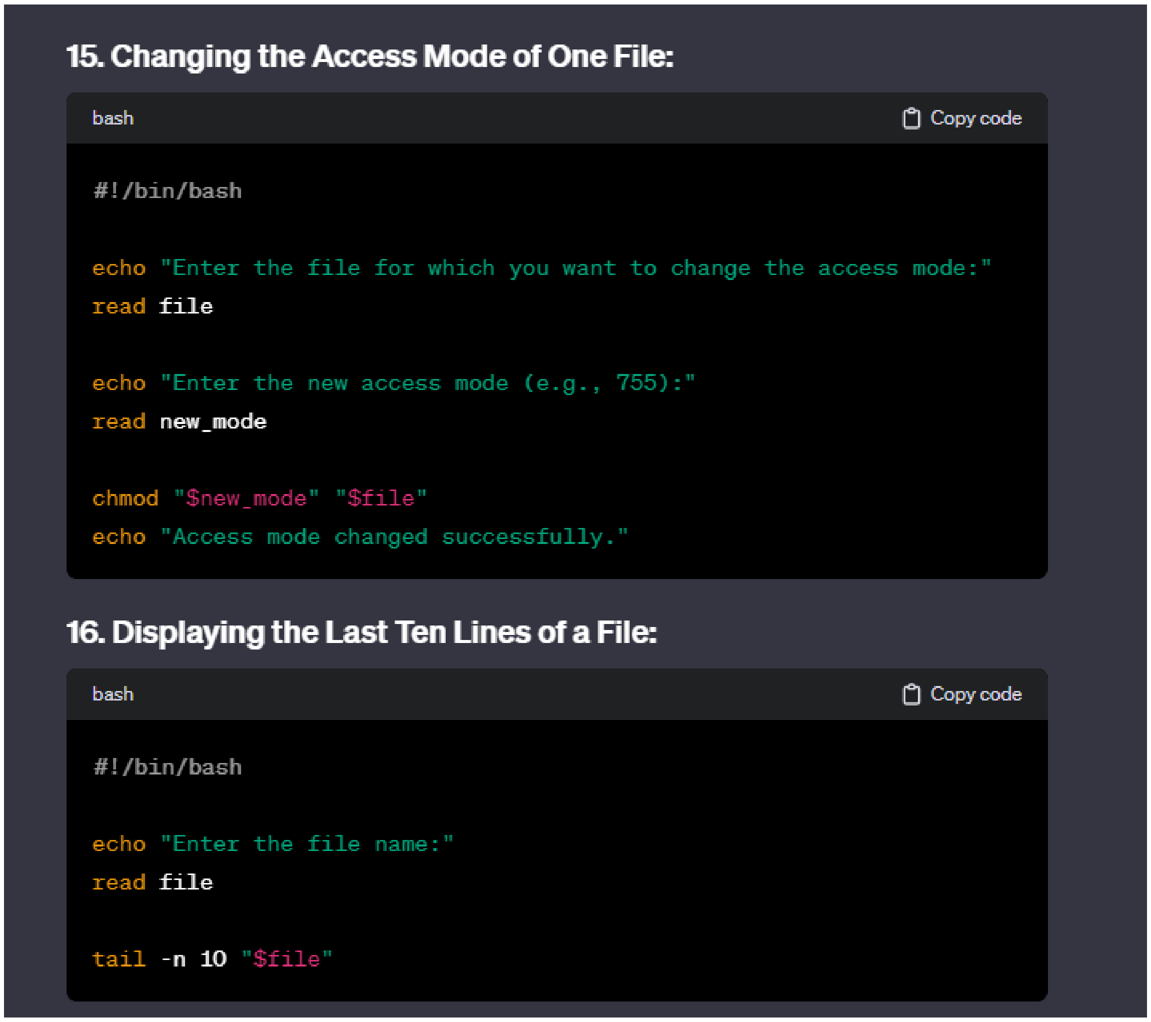
14. Merging Two Files into One:

bash Copy code

```bash
#!/bin/bash

echo "Enter the first file:"
read file1

echo "Enter the second file:"
read file2

echo "Enter the output file:"
read output_file

cat "$file1" "$file2" > "$output_file"
echo "Files merged successfully."
```

15. changes the access mode of one file

16. display the last ten lines of the file.



15. Changing the Access Mode of One File:

bash Copy code

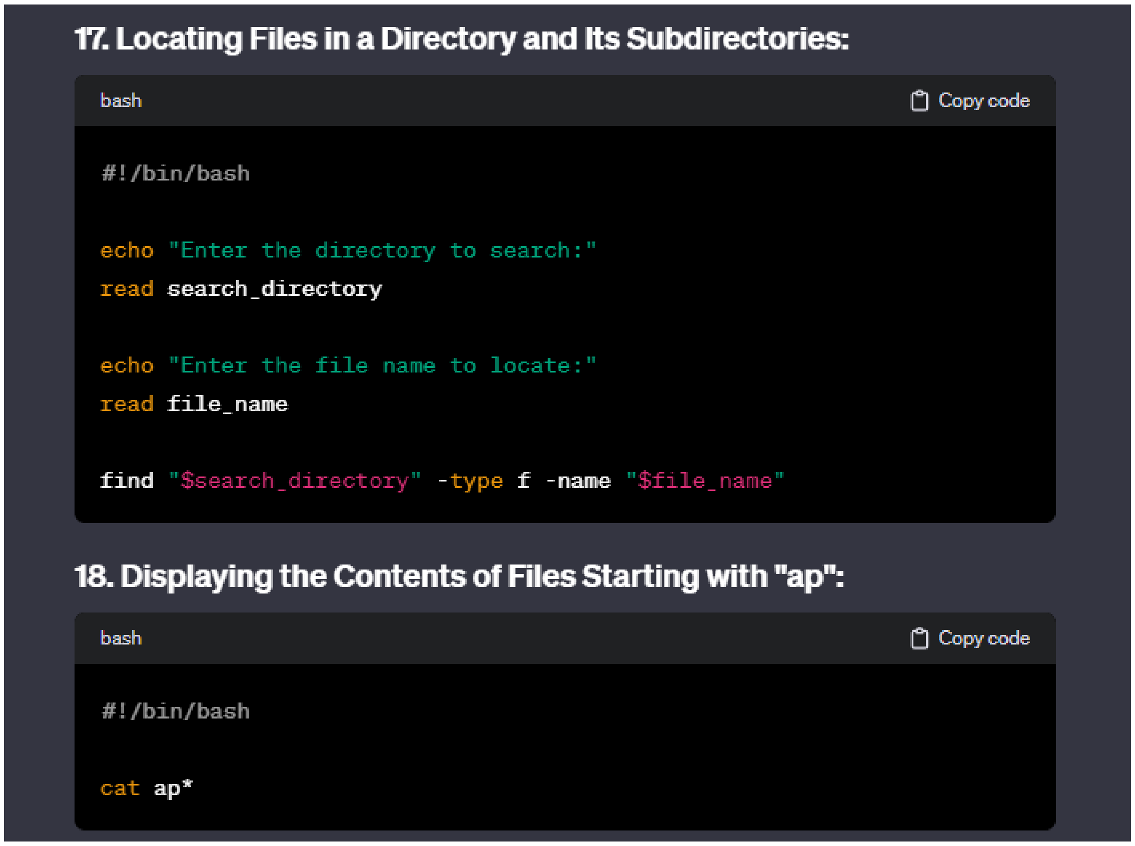
```bash
#!/bin/bash

echo "Enter the file for which you want to change the access mode:"
read file

echo "Enter the new access mode (e.g., 755):"
read new_mode

chmod "$new_mode" "$file"
echo "Access mode changed successfully."
```

16. Displaying the Last Ten Lines of a File:

bash Copy code

```bash
#!/bin/bash

echo "Enter the file name:"
read file

tail -n 10 "$file"
```

17. **to locate files in a directory and in a subdirectory.**
18. **This displays the contents of all files having a name starting with ap followed by any number of characters.**



17. Locating Files in a Directory and Its Subdirectories:

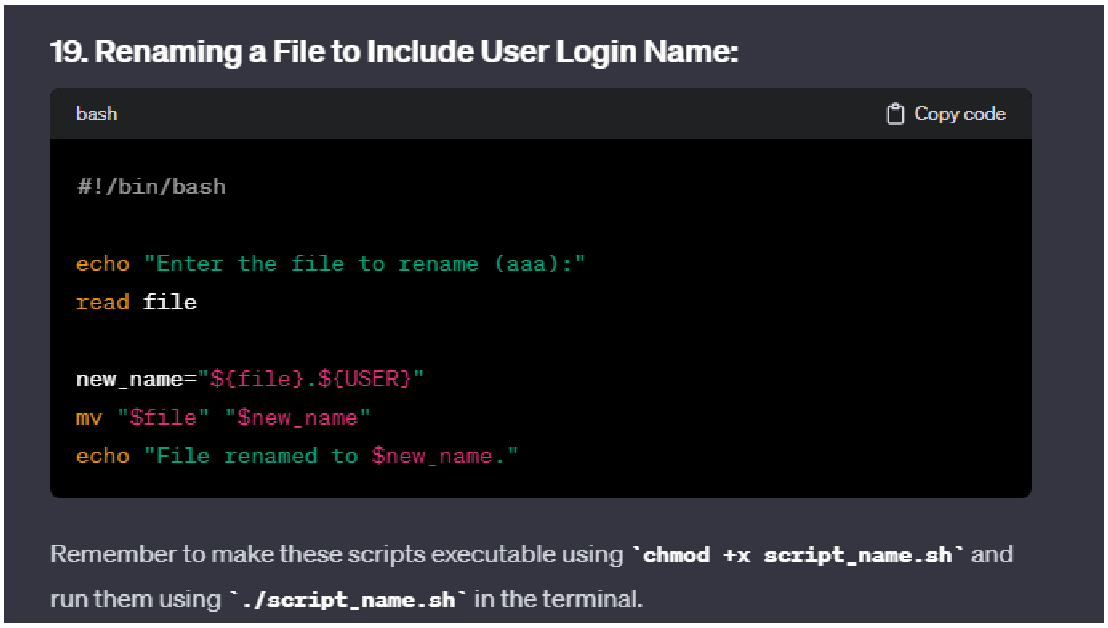
bash

```bash
#!/bin/bash

echo "Enter the directory to search:"
read search_directory

echo "Enter the file name to locate:"
read file_name

find "$search_directory" -type f -name "$file_name"
```

18. Displaying the Contents of Files Starting with "ap":

bash

```bash
#!/bin/bash

cat ap*
```



19. Renaming a File to Include User Login Name:

bash

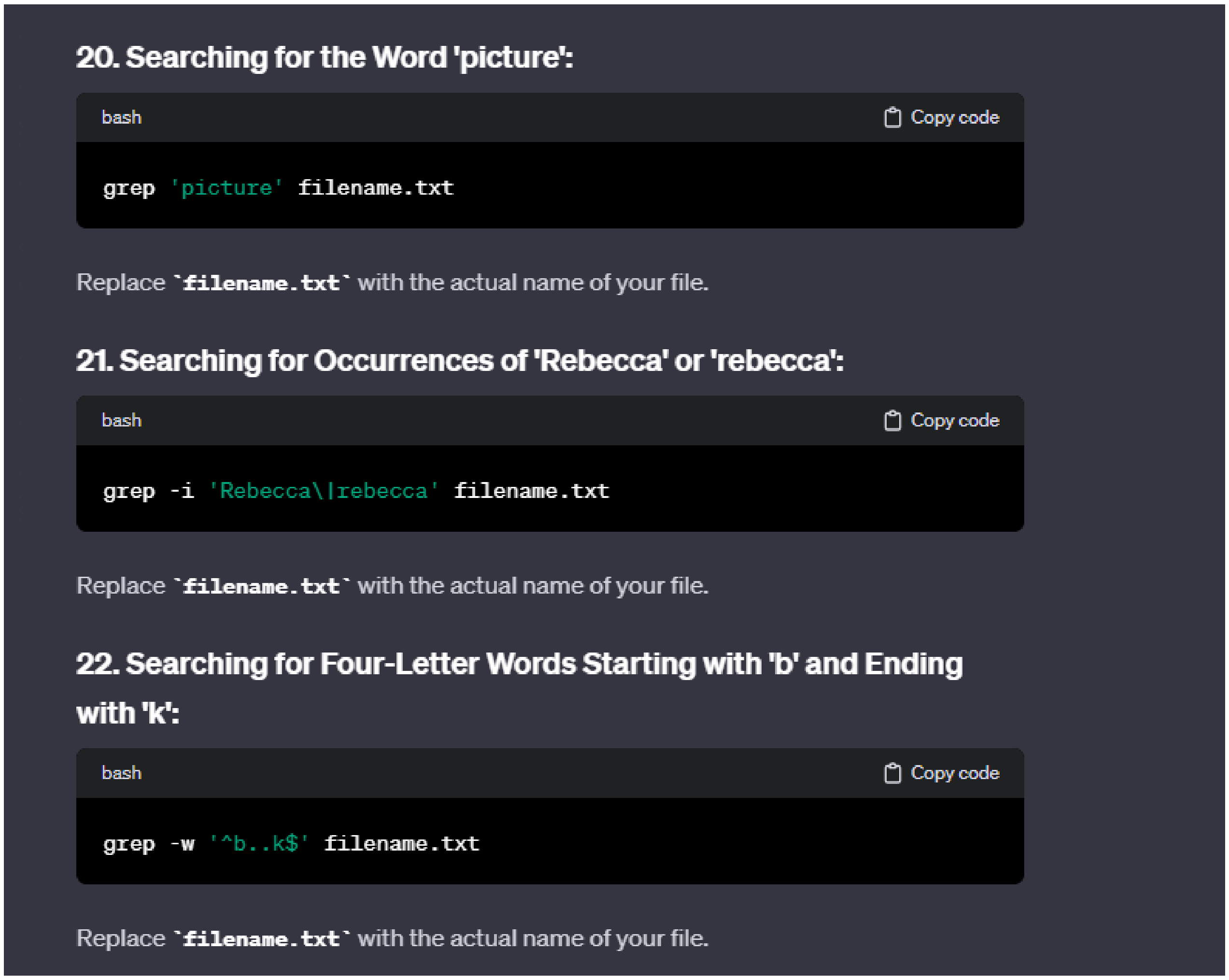
```bash
#!/bin/bash

echo "Enter the file to rename (aaa):"
read file

new_name="${file}.${USER}"
mv "$file" "$new_name"
echo "File renamed to $new_name."
```

Remember to make these scripts executable using `chmod +x script_name.sh` and run them using `./script_name.sh` in the terminal.

19. Rename any file aaa to aaa.aa1, where aa1 is the user login name.

Illustrate the use of sort, grep, awk, etc.

20. **Write a command to search the word 'picture' in the file and if found, the lines containing it would be displayed on the screen.**
21. **Write a command to search for all occurrences of 'Rebecca' as well as 'rebecca' in file and display the lines which contain one of these words.**



20. Searching for the Word 'picture':

```bash
grep 'picture' filename.txt
```

Replace `filename.txt` with the actual name of your file.

21. Searching for Occurrences of 'Rebecca' or 'rebecca':

```bash
grep -i 'Rebecca\|rebecca' filename.txt
```

Replace `filename.txt` with the actual name of your file.

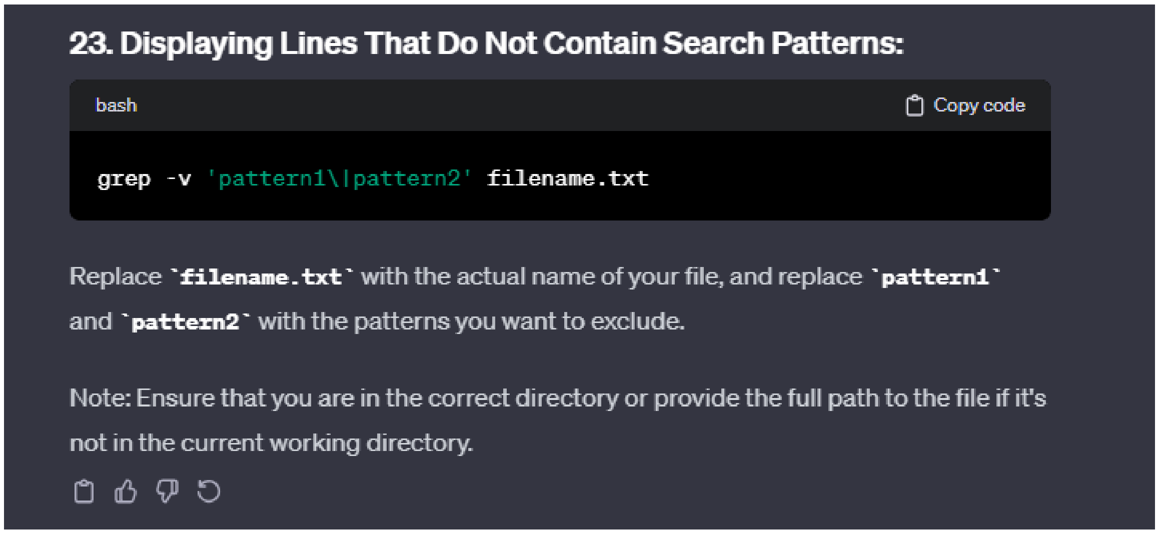
22. Searching for Four-Letter Words Starting with 'b' and Ending with 'k':

```bash
grep -w '^b..k$' filename.txt
```

Replace `filename.txt` with the actual name of your file.

22. **Write a command to search all four-letter words whose first letter is a 'b' and last letter, a 'k'.**

23. **Write a command to see only those lines which do not contain the search patterns**



23. Displaying Lines That Do Not Contain Search Patterns:

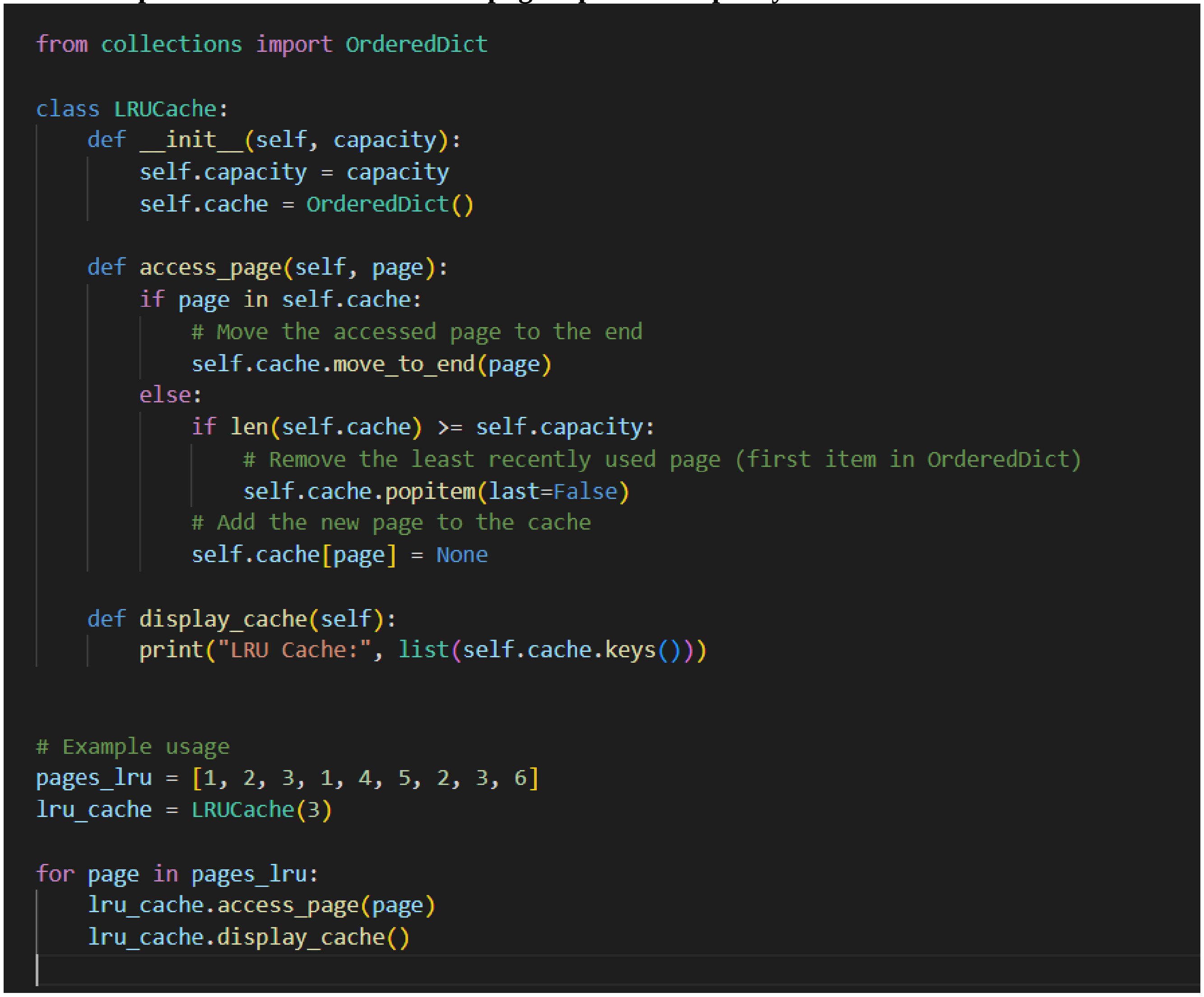
```bash
grep -v 'pattern1\|pattern2' filename.txt
```

Replace `filename.txt` with the actual name of your file, and replace `pattern1` and `pattern2` with the patterns you want to exclude.

Note: Ensure that you are in the correct directory or provide the full path to the file if it's not in the current working directory.

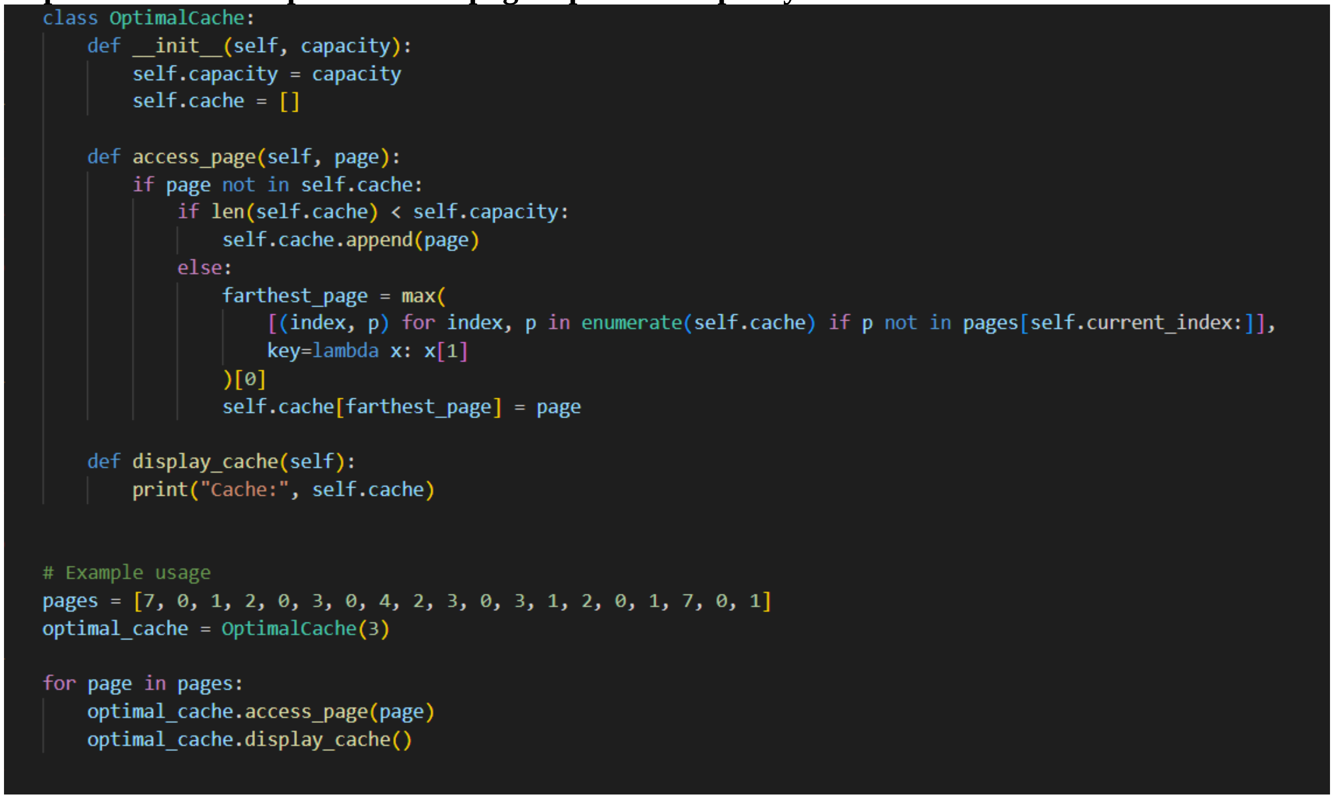
24. Implement Booth's multiplication algorithm.
25. Implement Restoring division algorithm.
26. Implement Non-Restoring division algorithm.
27. Implement fully associative memory mapped cache organization.

28. Implement various LRU cache/page replacement policy



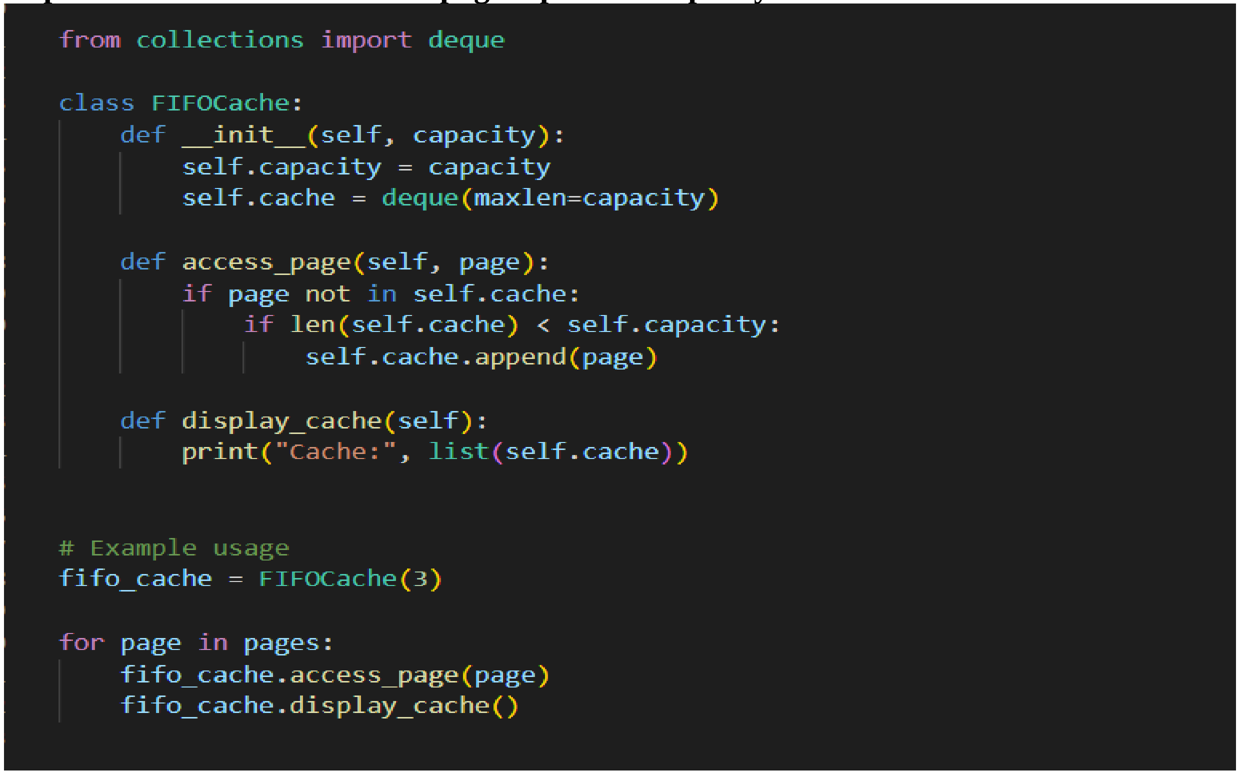
```python
from collections import OrderedDict

class LRUCache:
    def __init__(self, capacity):
        self.capacity = capacity
        self.cache = OrderedDict()

    def access_page(self, page):
        if page in self.cache:
            # Move the accessed page to the end
            self.cache.move_to_end(page)
        else:
            if len(self.cache) >= self.capacity:
                # Remove the least recently used page (first item in OrderedDict)
                self.cache.popitem(last=False)
            # Add the new page to the cache
            self.cache[page] = None

    def display_cache(self):
        print("LRU Cache:", list(self.cache.keys()))


# Example usage
pages_lru = [1, 2, 3, 1, 4, 5, 2, 3, 6]
lru_cache = LRUCache(3)

for page in pages_lru:
    lru_cache.access_page(page)
    lru_cache.display_cache()
```

29. Implement various optimal cache/page replacement policy



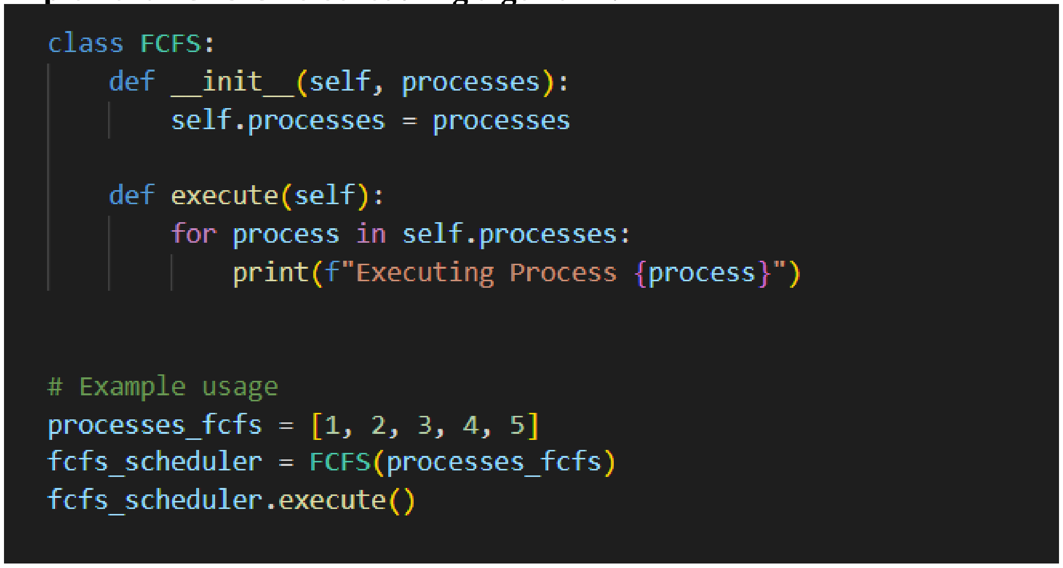
```python
class OptimalCache:
    def __init__(self, capacity):
        self.capacity = capacity
        self.cache = []

    def access_page(self, page):
        if page not in self.cache:
            if len(self.cache) < self.capacity:
                self.cache.append(page)
            else:
                farthest_page = max(
                    [(index, p) for index, p in enumerate(self.cache) if p not in pages[self.current_index:]],
                    key=lambda x: x[1]
                )[0]
                self.cache[farthest_page] = page

    def display_cache(self):
        print("Cache:", self.cache)


# Example usage
pages = [7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 1, 2, 0, 1, 7, 0, 1]
optimal_cache = OptimalCache(3)

for page in pages:
    optimal_cache.access_page(page)
    optimal_cache.display_cache()
```

30. Implement various FIFO cache/page replacement policy



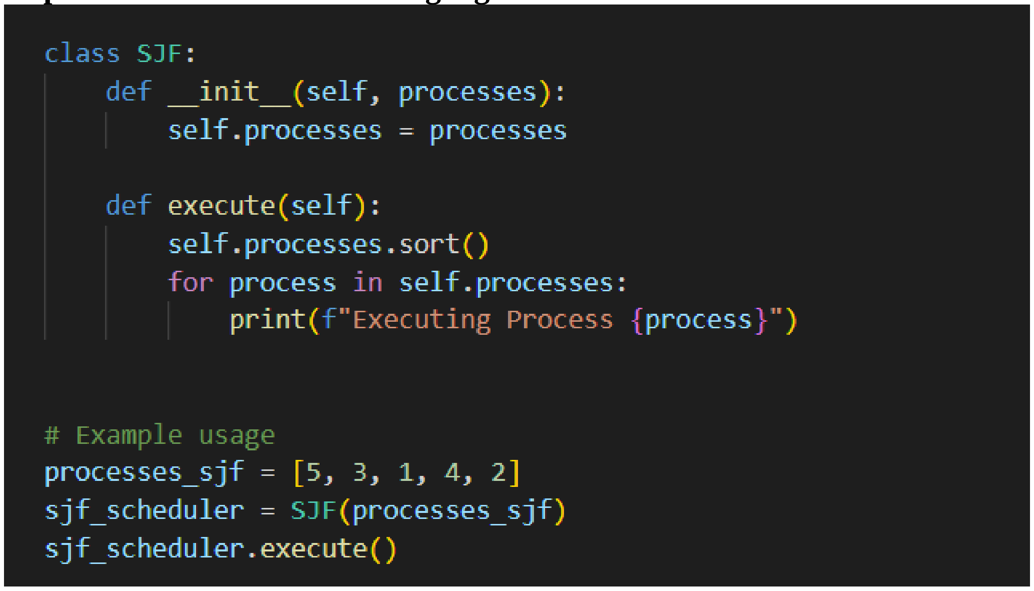
```python
from collections import deque

class FIFOCache:
    def __init__(self, capacity):
        self.capacity = capacity
        self.cache = deque(maxlen=capacity)

    def access_page(self, page):
        if page not in self.cache:
            if len(self.cache) < self.capacity:
                self.cache.append(page)

    def display_cache(self):
        print("Cache:", list(self.cache))


# Example usage
fifo_cache = FIFOCache(3)

for page in pages:
    fifo_cache.access_page(page)
    fifo_cache.display_cache()
```

31. Implement FCFS CPU scheduling algorithm.



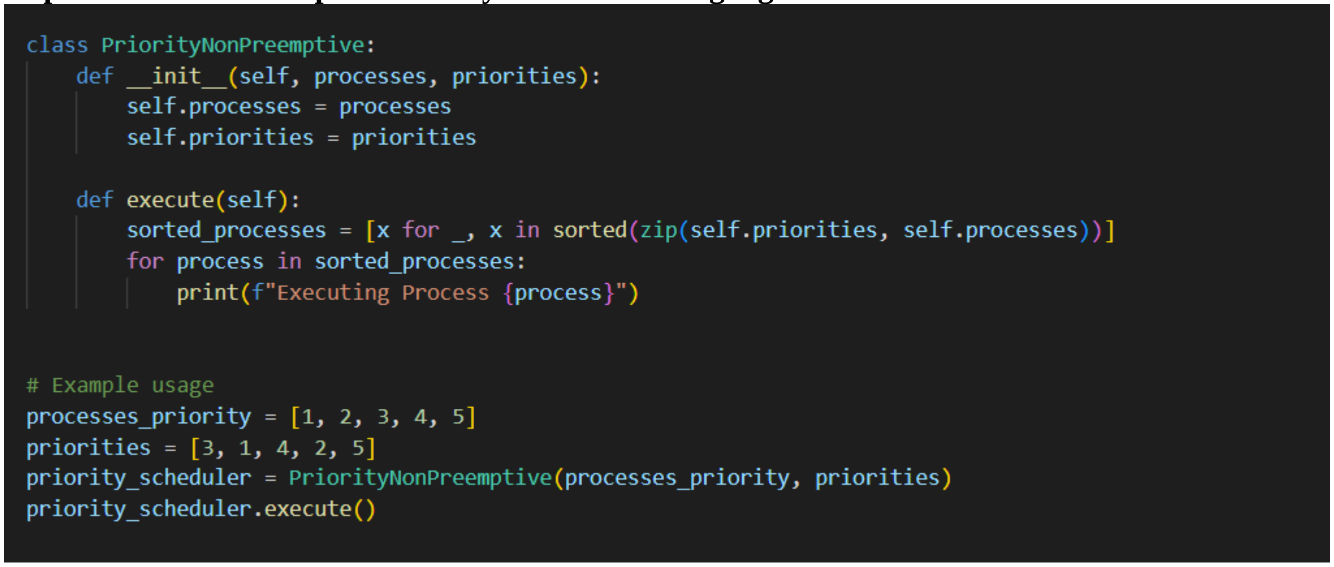
```
class FCFS:
    def __init__(self, processes):
        self.processes = processes

    def execute(self):
        for process in self.processes:
            print(f"Executing Process {process}")


# Example usage
processes_fcfs = [1, 2, 3, 4, 5]
fcfs_scheduler = FCFS(processes_fcfs)
fcfs_scheduler.execute()
```

32. Implement SJF CPU scheduling algorithm.



```
class SJF:
    def __init__(self, processes):
        self.processes = processes

    def execute(self):
        self.processes.sort()
        for process in self.processes:
            print(f"Executing Process {process}")


# Example usage
processes_sjf = [5, 3, 1, 4, 2]
sjf_scheduler = SJF(processes_sjf)
sjf_scheduler.execute()
```

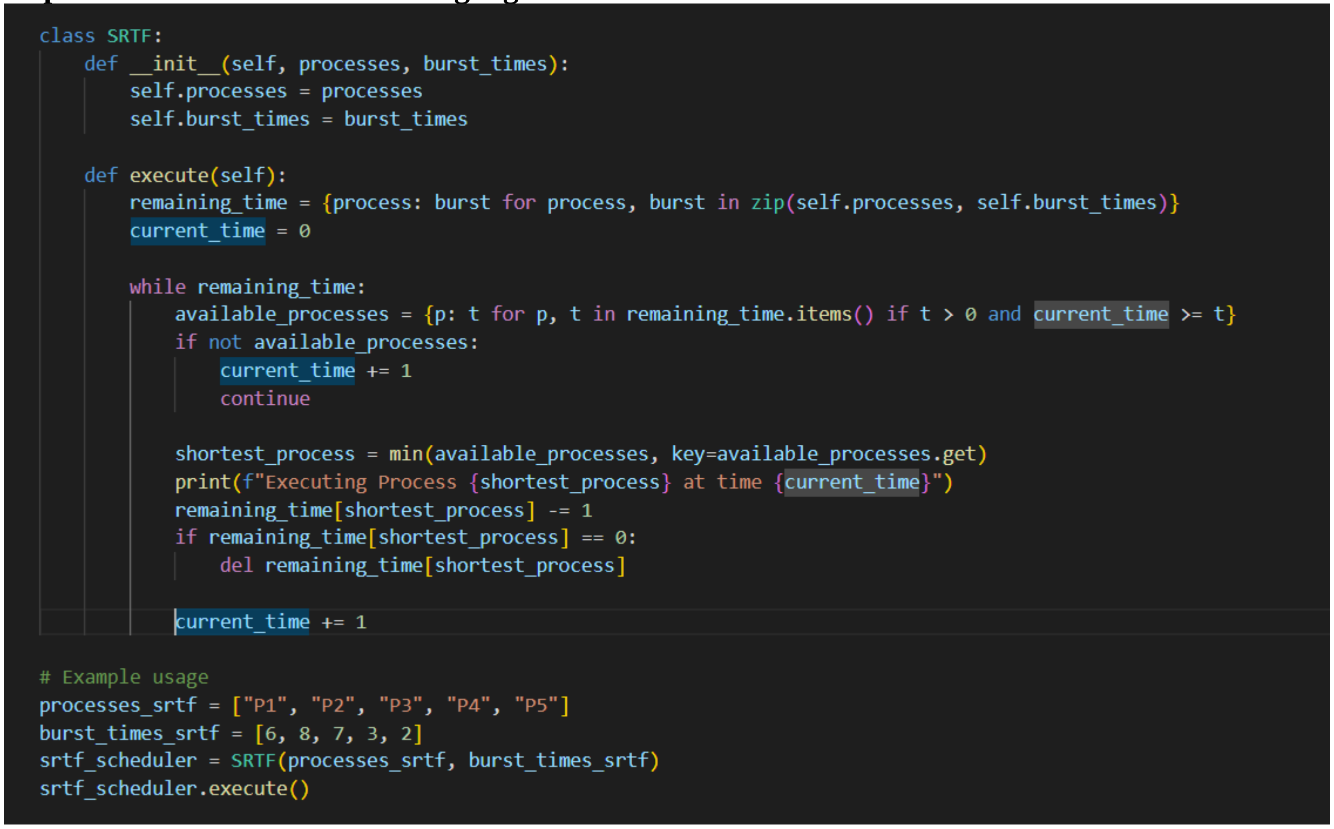
33. Implement Non Prremptive Priority CPU scheduling algorithm.



```python
class PriorityNonPreemptive:
    def __init__(self, processes, priorities):
        self.processes = processes
        self.priorities = priorities

    def execute(self):
        sorted_processes = [x for _, x in sorted(zip(self.priorities, self.processes))]
        for process in sorted_processes:
            print(f"Executing Process {process}")


# Example usage
processes_priority = [1, 2, 3, 4, 5]
priorities = [3, 1, 4, 2, 5]
priority_scheduler = PriorityNonPreemptive(processes_priority, priorities)
priority_scheduler.execute()
```

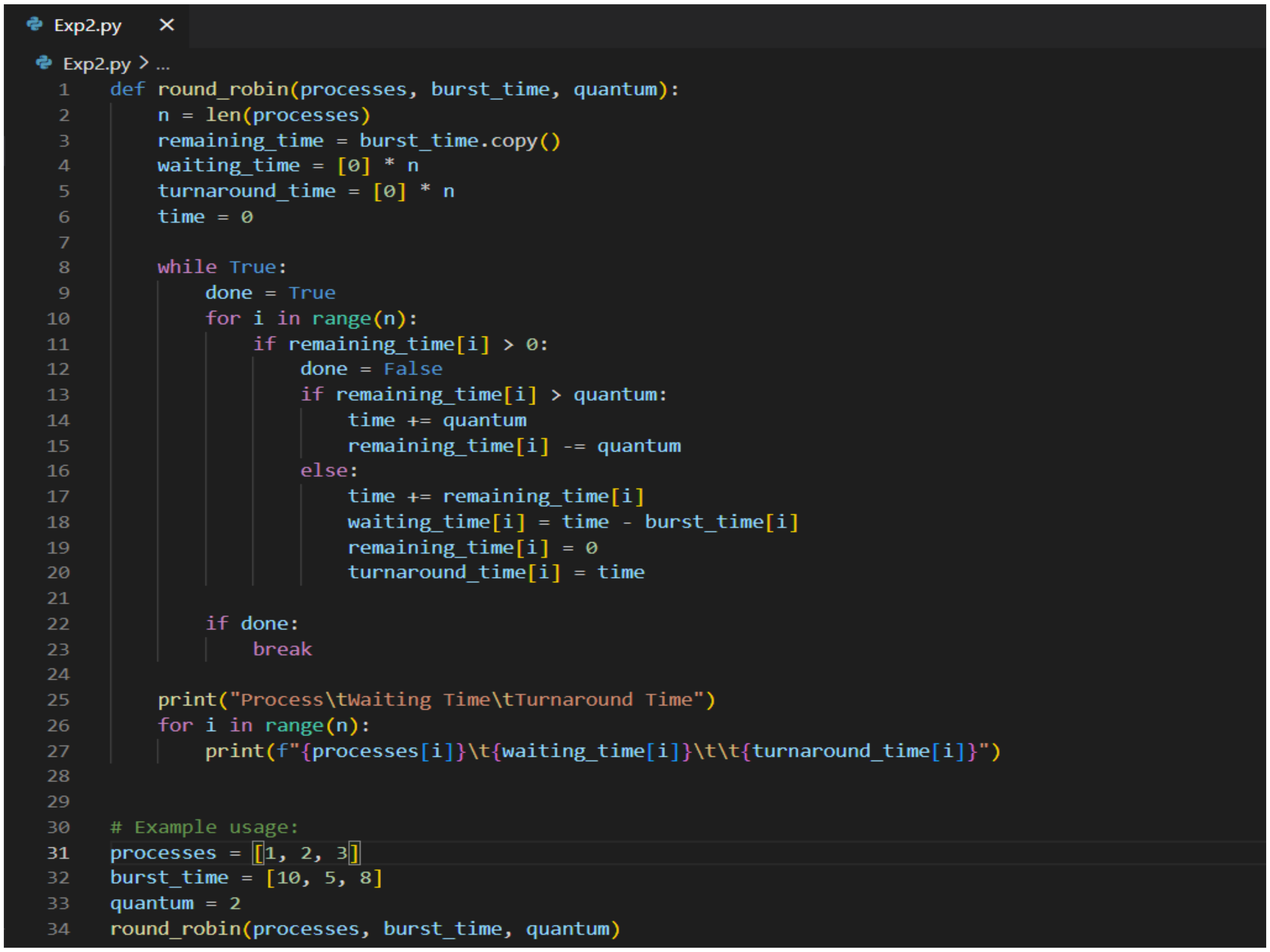
34. Implement Prremptive Priority CPU scheduling algorithm.

35. Implement SRTF CPU scheduling algorithm.



```python
class SRTF:
    def __init__(self, processes, burst_times):
        self.processes = processes
        self.burst_times = burst_times

    def execute(self):
        remaining_time = {process: burst for process, burst in zip(self.processes, self.burst_times)}
        current_time = 0

        while remaining_time:
            available_processes = {p: t for p, t in remaining_time.items() if t > 0 and current_time >= t}
            if not available_processes:
                current_time += 1
                continue

            shortest_process = min(available_processes, key=available_processes.get)
            print(f"Executing Process {shortest_process} at time {current_time}")
            remaining_time[shortest_process] -= 1
            if remaining_time[shortest_process] == 0:
                del remaining_time[shortest_process]

            current_time += 1

# Example usage
processes_srtf = ["P1", "P2", "P3", "P4", "P5"]
burst_times_srtf = [6, 8, 7, 3, 2]
srtf_scheduler = SRTF(processes_srtf, burst_times_srtf)
srtf_scheduler.execute()
```

36. Implement Round Robin CPU scheduling algorithm.

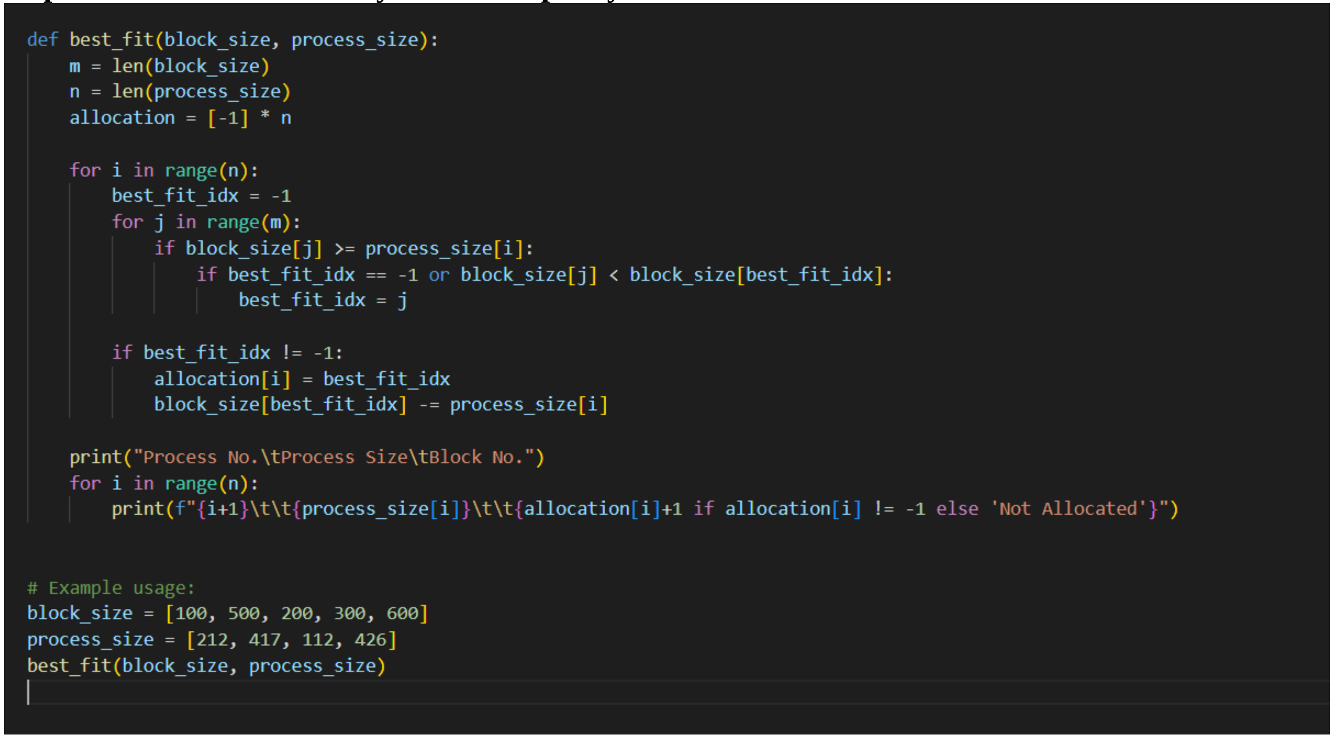


Exp2.py

Exp2.py > ...

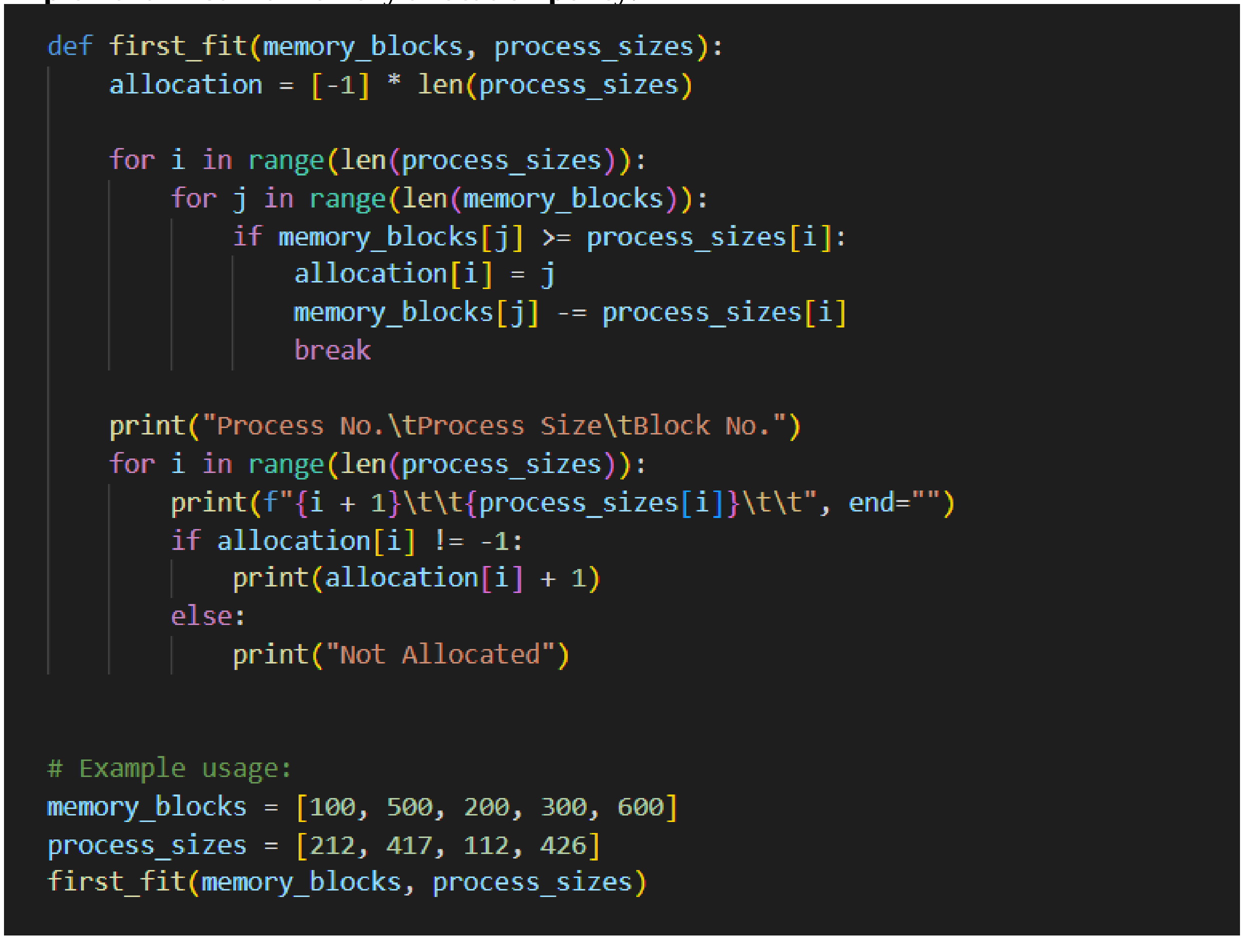
```python
def round_robin(processes, burst_time, quantum):
    n = len(processes)
    remaining_time = burst_time.copy()
    waiting_time = [0] * n
    turnaround_time = [0] * n
    time = 0

    while True:
        done = True
        for i in range(n):
            if remaining_time[i] > 0:
                done = False
                if remaining_time[i] > quantum:
                    time += quantum
                    remaining_time[i] -= quantum
                else:
                    time += remaining_time[i]
                    waiting_time[i] = time - burst_time[i]
                    remaining_time[i] = 0
                    turnaround_time[i] = time

        if done:
            break

    print("Process\tWaiting Time\tTurnaround Time")
    for i in range(n):
        print(f"{processes[i]}\t{waiting_time[i]}\t\t{turnaround_time[i]}")


# Example usage:
processes = [1, 2, 3]
burst_time = [10, 5, 8]
quantum = 2
round_robin(processes, burst_time, quantum)
```

37. Implement Best Fit Memory allocation policy.



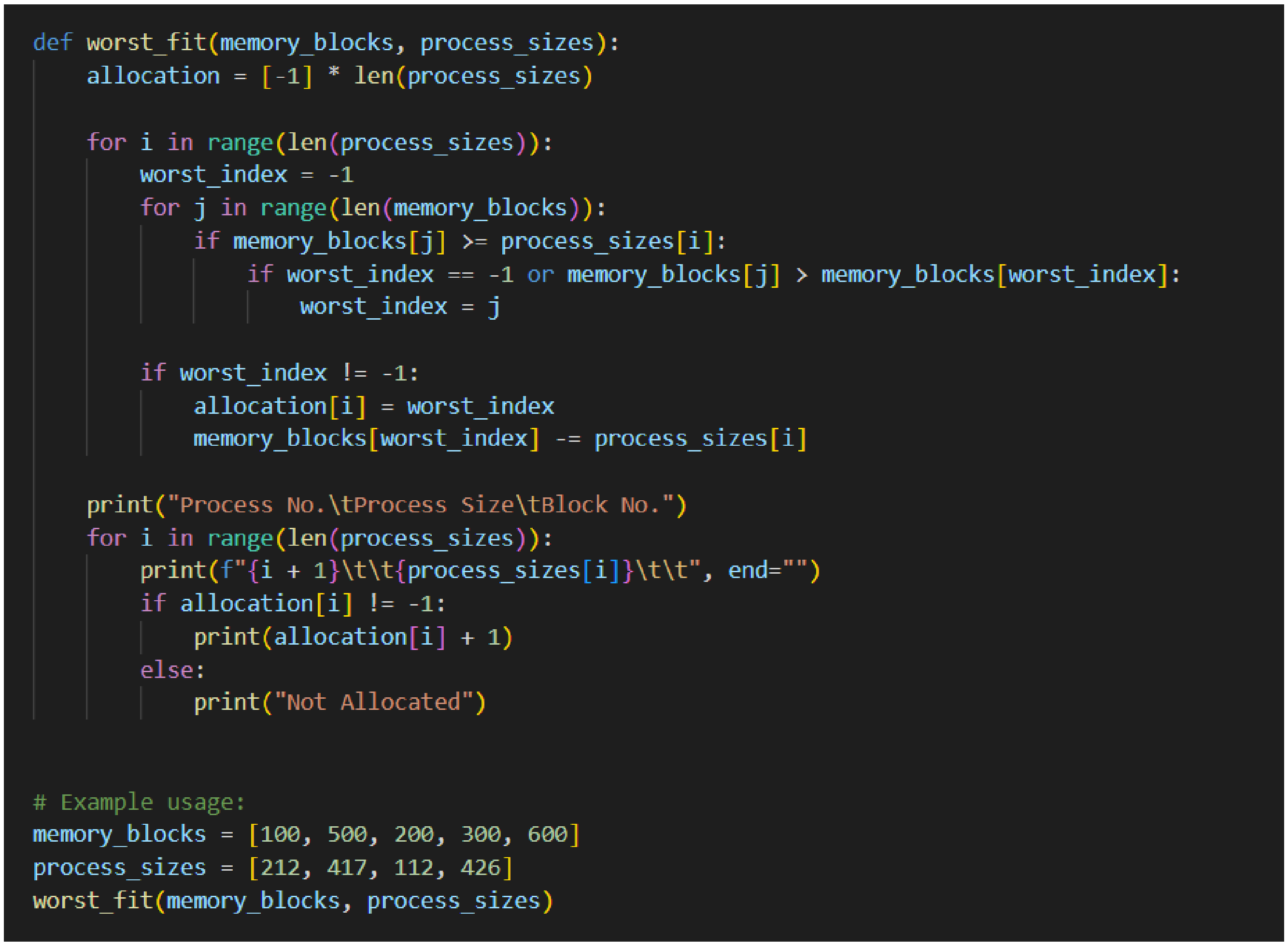
```python
def best_fit(block_size, process_size):
    m = len(block_size)
    n = len(process_size)
    allocation = [-1] * n

    for i in range(n):
        best_fit_idx = -1
        for j in range(m):
            if block_size[j] >= process_size[i]:
                if best_fit_idx == -1 or block_size[j] < block_size[best_fit_idx]:
                    best_fit_idx = j

        if best_fit_idx != -1:
            allocation[i] = best_fit_idx
            block_size[best_fit_idx] -= process_size[i]

    print("Process No.\tProcess Size\tBlock No.")
    for i in range(n):
        print(f"{i+1}\t\t{process_size[i]}\t\t{allocation[i]+1 if allocation[i] != -1 else 'Not Allocated'}")


# Example usage:
block_size = [100, 500, 200, 300, 600]
process_size = [212, 417, 112, 426]
best_fit(block_size, process_size)
```

38. Implement First Fit Memory allocation policy.



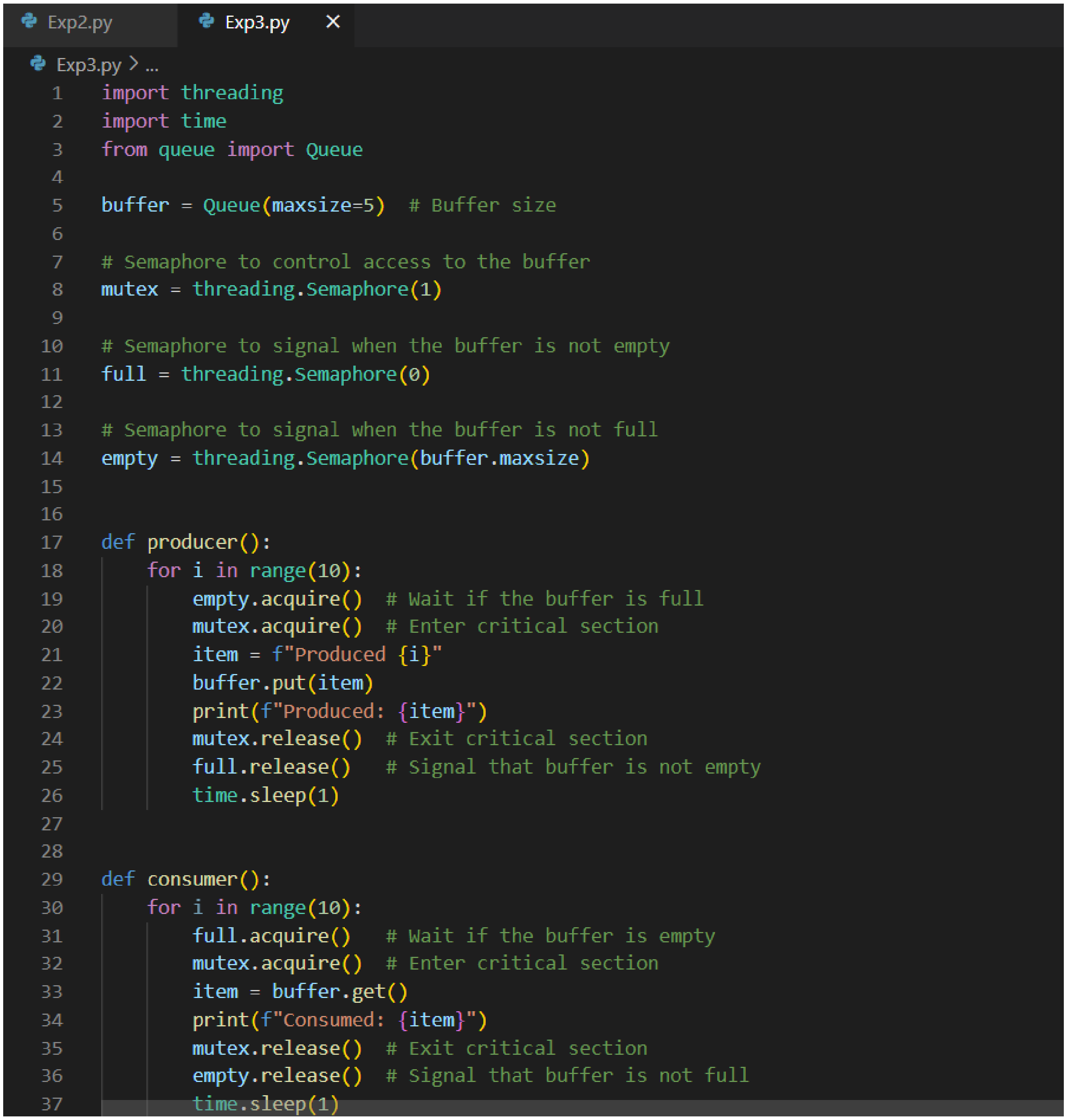
```
def first_fit(memory_blocks, process_sizes):
    allocation = [-1] * len(process_sizes)

    for i in range(len(process_sizes)):
        for j in range(len(memory_blocks)):
            if memory_blocks[j] >= process_sizes[i]:
                allocation[i] = j
                memory_blocks[j] -= process_sizes[i]
                break

    print("Process No.\tProcess Size\tBlock No.")
    for i in range(len(process_sizes)):
        print(f"{i + 1}\t\t{process_sizes[i]}\t\t", end="")
        if allocation[i] != -1:
            print(allocation[i] + 1)
        else:
            print("Not Allocated")


# Example usage:
memory_blocks = [100, 500, 200, 300, 600]
process_sizes = [212, 417, 112, 426]
first_fit(memory_blocks, process_sizes)
```

39. Implement Worst Fit Memory allocation policy.



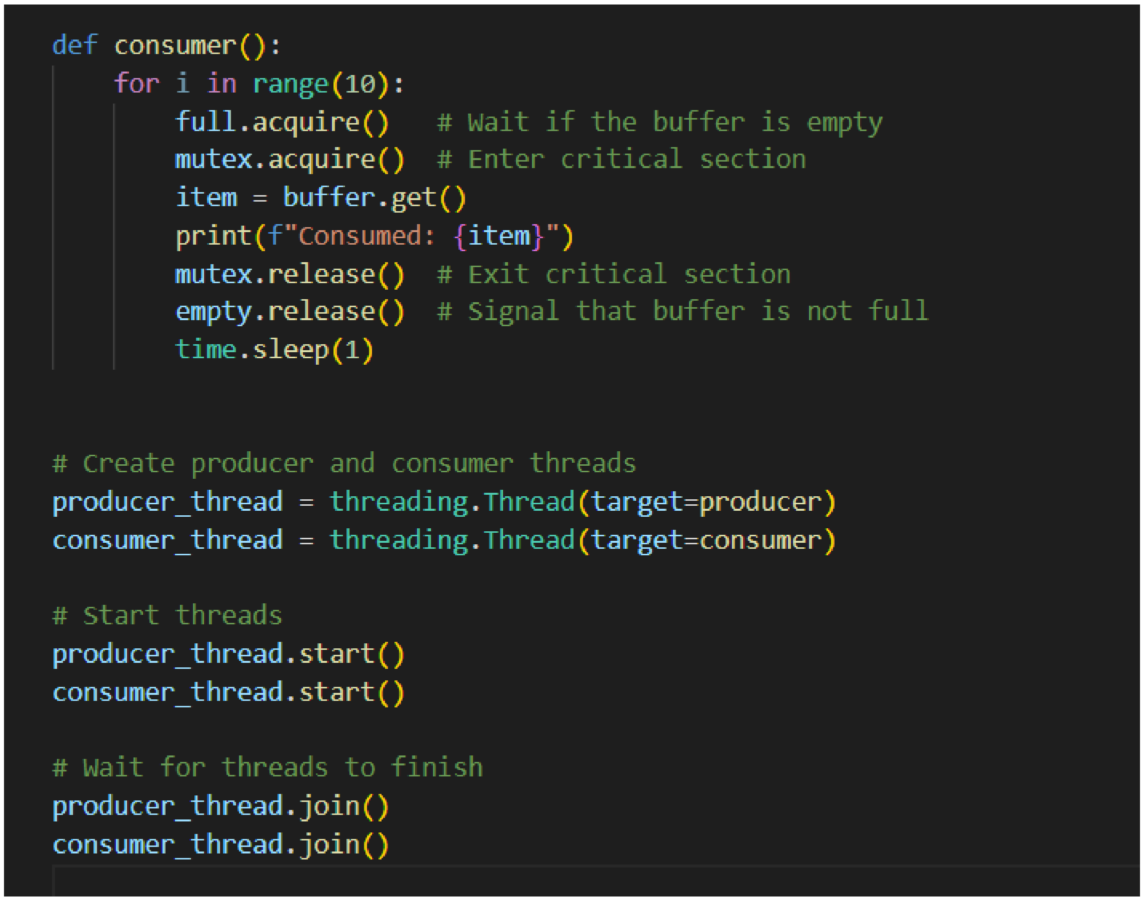
```python
def worst_fit(memory_blocks, process_sizes):
    allocation = [-1] * len(process_sizes)

    for i in range(len(process_sizes)):
        worst_index = -1
        for j in range(len(memory_blocks)):
            if memory_blocks[j] >= process_sizes[i]:
                if worst_index == -1 or memory_blocks[j] > memory_blocks[worst_index]:
                    worst_index = j

        if worst_index != -1:
            allocation[i] = worst_index
            memory_blocks[worst_index] -= process_sizes[i]

    print("Process No.\tProcess Size\tBlock No.")
    for i in range(len(process_sizes)):
        print(f"{i + 1}\t\t{process_sizes[i]}\t\t", end="")
        if allocation[i] != -1:
            print(allocation[i] + 1)
        else:
            print("Not Allocated")


# Example usage:
memory_blocks = [100, 500, 200, 300, 600]
process_sizes = [212, 417, 112, 426]
worst_fit(memory_blocks, process_sizes)
```

40. Implement Producer -Consumer problem with Semaphore.


Exp2.py Exp3.py ×

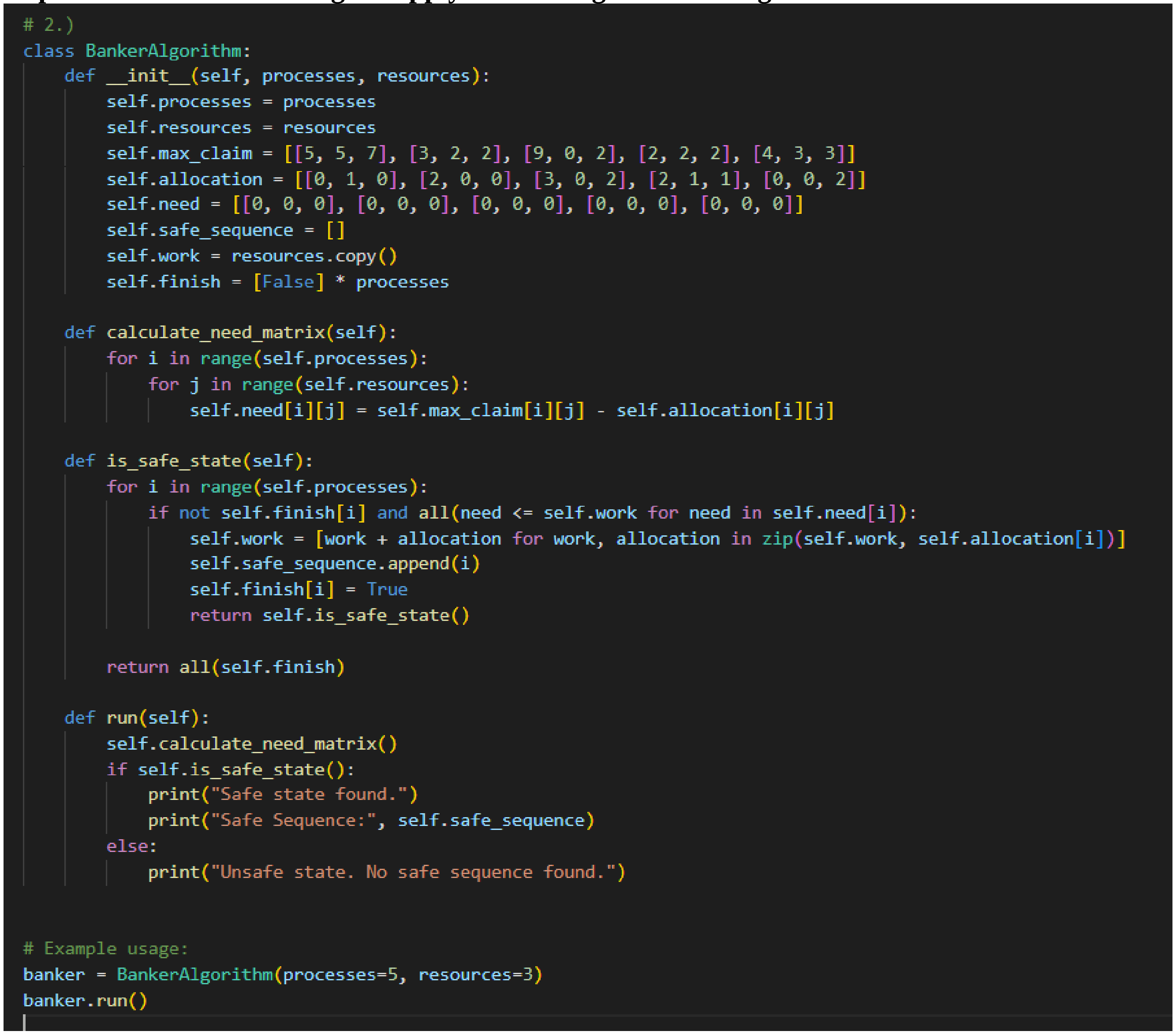
Exp3.py > ...

```python
import threading
import time
from queue import Queue

buffer = Queue(maxsize=5)  # Buffer size

# Semaphore to control access to the buffer
mutex = threading.Semaphore(1)

# Semaphore to signal when the buffer is not empty
full = threading.Semaphore(0)

# Semaphore to signal when the buffer is not full
empty = threading.Semaphore(buffer.maxsize)


def producer():
    for i in range(10):
        empty.acquire()  # Wait if the buffer is full
        mutex.acquire()  # Enter critical section
        item = f"Produced {i}"
        buffer.put(item)
        print(f"Produced: {item}")
        mutex.release()  # Exit critical section
        full.release()   # Signal that buffer is not empty
        time.sleep(1)


def consumer():
    for i in range(10):
        full.acquire()   # Wait if the buffer is empty
        mutex.acquire()  # Enter critical section
        item = buffer.get()
        print(f"Consumed: {item}")
        mutex.release()  # Exit critical section
        empty.release()  # Signal that buffer is not full
        time.sleep(1)
```



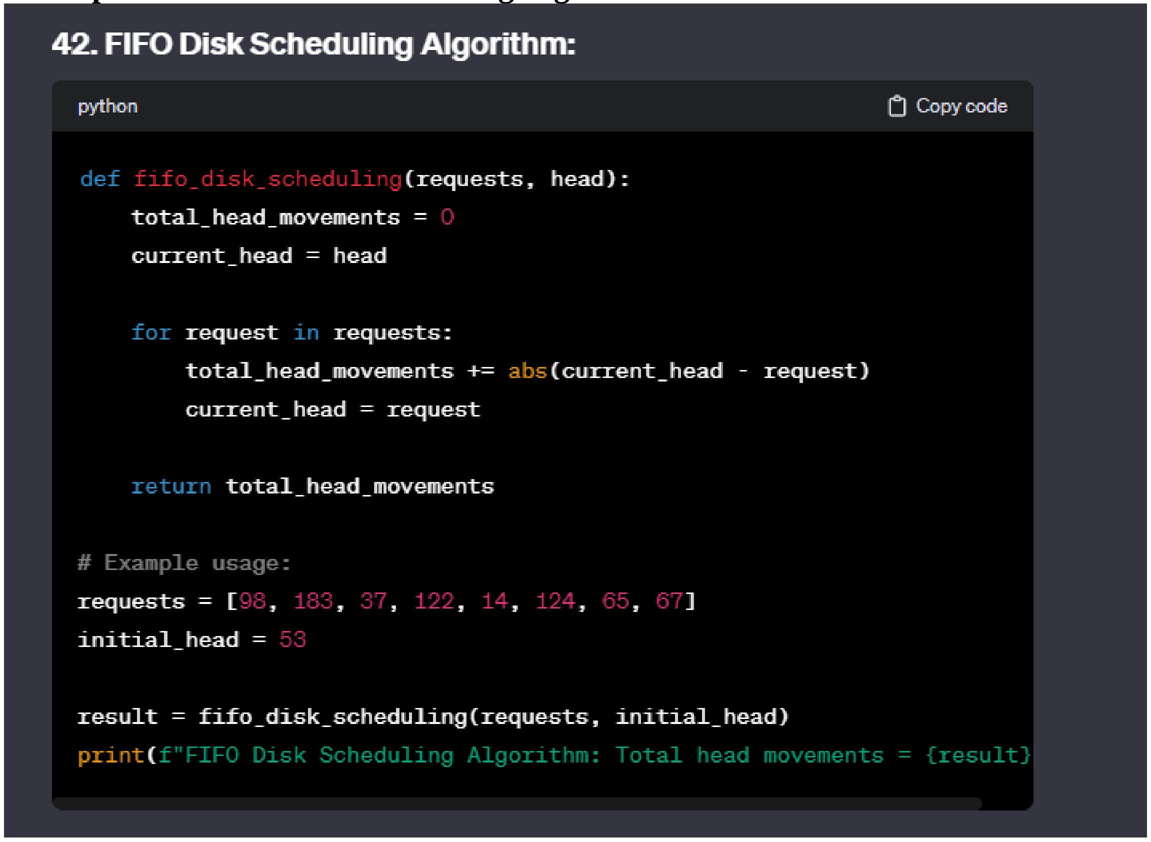
```
def consumer():
    for i in range(10):
        full.acquire()   # Wait if the buffer is empty
        mutex.acquire()  # Enter critical section
        item = buffer.get()
        print(f"Consumed: {item}")
        mutex.release()  # Exit critical section
        empty.release()  # Signal that buffer is not full
        time.sleep(1)


# Create producer and consumer threads
producer_thread = threading.Thread(target=producer)
consumer_thread = threading.Thread(target=consumer)

# Start threads
producer_thread.start()
consumer_thread.start()

# Wait for threads to finish
producer_thread.join()
consumer_thread.join()
```

41. Implement order scheduling in supply chain using Banker's Algorithm



```python
# 2.)
class BankerAlgorithm:
    def __init__(self, processes, resources):
        self.processes = processes
        self.resources = resources
        self.max_claim = [[5, 5, 7], [3, 2, 2], [9, 0, 2], [2, 2, 2], [4, 3, 3]]
        self.allocation = [[0, 1, 0], [2, 0, 0], [3, 0, 2], [2, 1, 1], [0, 0, 2]]
        self.need = [[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]]
        self.safe_sequence = []
        self.work = resources.copy()
        self.finish = [False] * processes

    def calculate_need_matrix(self):
        for i in range(self.processes):
            for j in range(self.resources):
                self.need[i][j] = self.max_claim[i][j] - self.allocation[i][j]

    def is_safe_state(self):
        for i in range(self.processes):
            if not self.finish[i] and all(need <= self.work for need in self.need[i]):
                self.work = [work + allocation for work, allocation in zip(self.work, self.allocation[i])]
                self.safe_sequence.append(i)
                self.finish[i] = True
                return self.is_safe_state()

        return all(self.finish)

    def run(self):
        self.calculate_need_matrix()
        if self.is_safe_state():
            print("Safe state found.")
            print("Safe Sequence:", self.safe_sequence)
        else:
            print("Unsafe state. No safe sequence found.")


# Example usage:
banker = BankerAlgorithm(processes=5, resources=3)
banker.run()
```

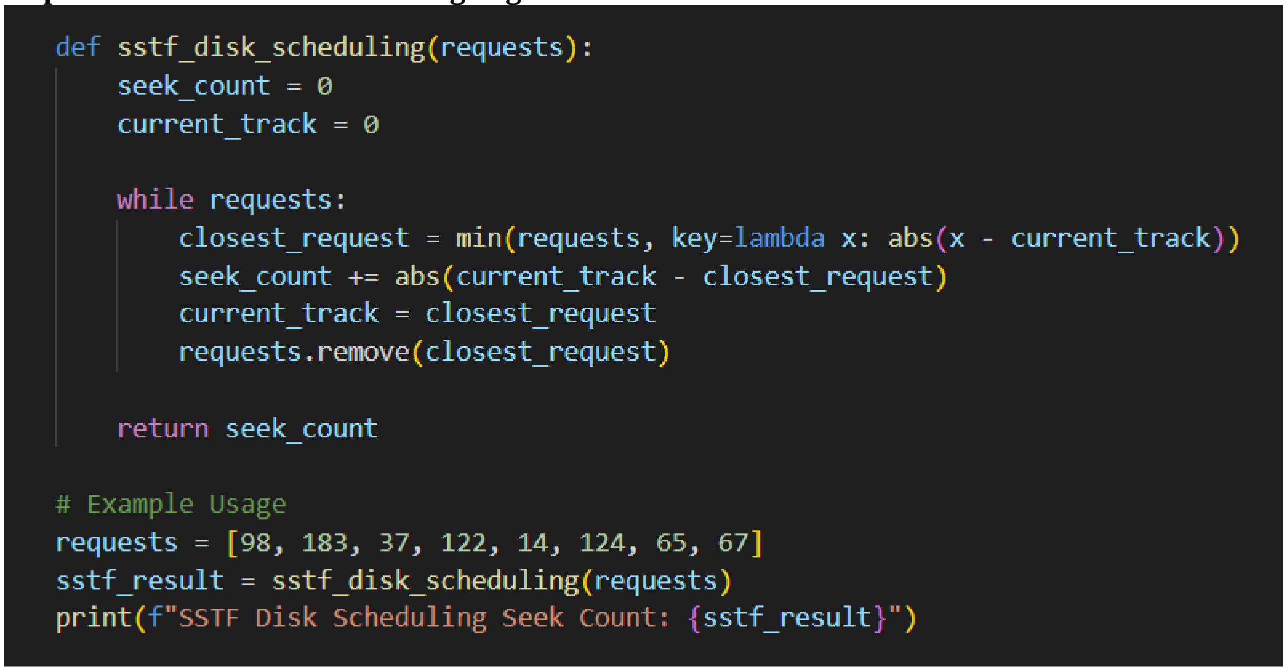
42. Implement FIFO Disk Scheduling Algorithms.



42. FIFO Disk Scheduling Algorithm:

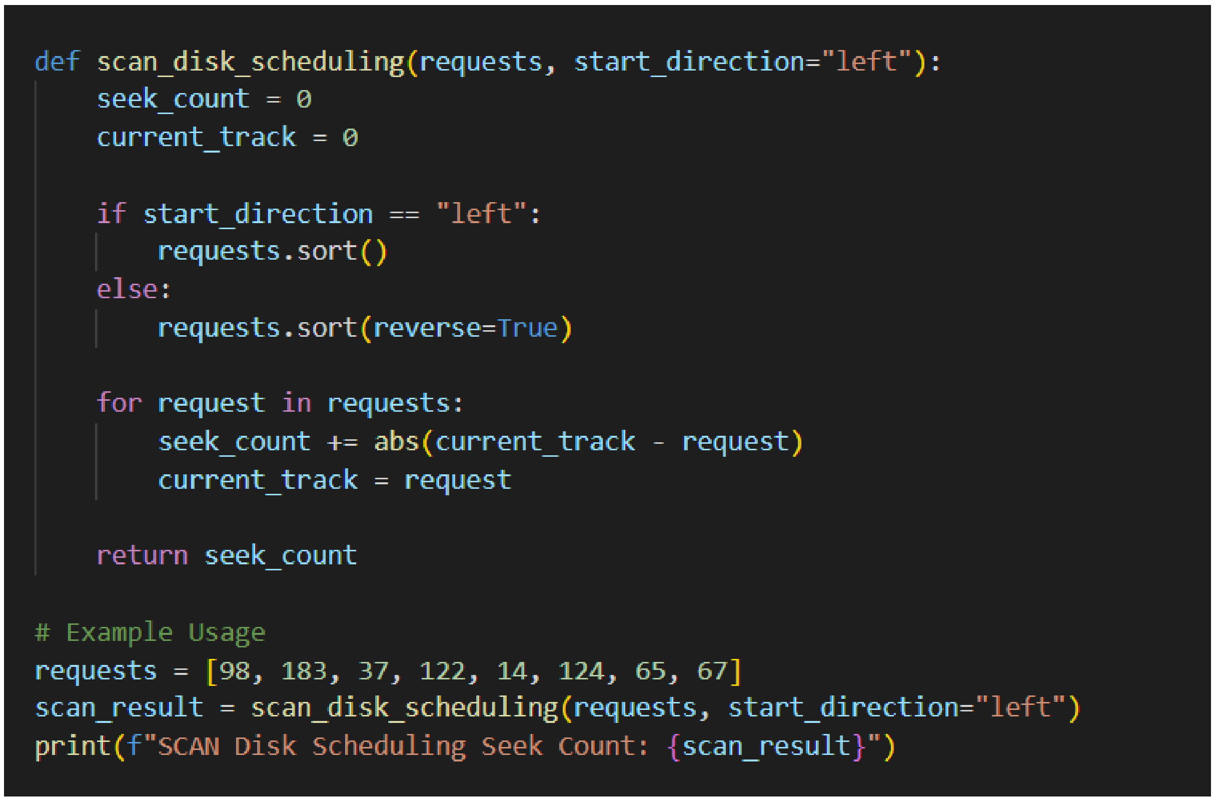
```python
def fifo_disk_scheduling(requests, head):
    total_head_movements = 0
    current_head = head

    for request in requests:
        total_head_movements += abs(current_head - request)
        current_head = request

    return total_head_movements

# Example usage:
requests = [98, 183, 37, 122, 14, 124, 65, 67]
initial_head = 53

result = fifo_disk_scheduling(requests, initial_head)
print(f"FIFO Disk Scheduling Algorithm: Total head movements = {result}
```

43. Implement SSTF Disk Scheduling Algorithms.



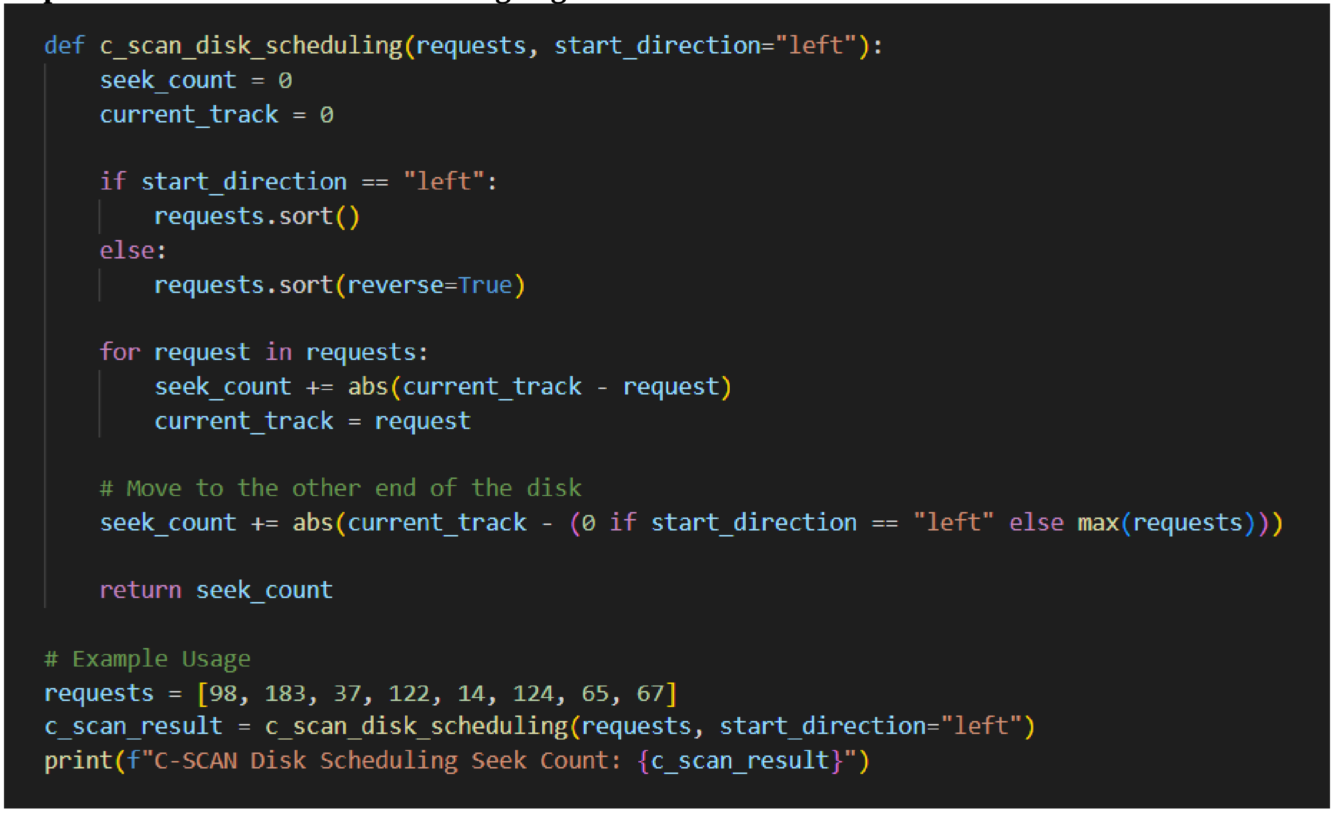
```python
def sstf_disk_scheduling(requests):
    seek_count = 0
    current_track = 0

    while requests:
        closest_request = min(requests, key=lambda x: abs(x - current_track))
        seek_count += abs(current_track - closest_request)
        current_track = closest_request
        requests.remove(closest_request)

    return seek_count

# Example Usage
requests = [98, 183, 37, 122, 14, 124, 65, 67]
sstf_result = sstf_disk_scheduling(requests)
print(f"SSTF Disk Scheduling Seek Count: {sstf_result}")
```

44. Implement SCAN Disk Scheduling Algorithms.



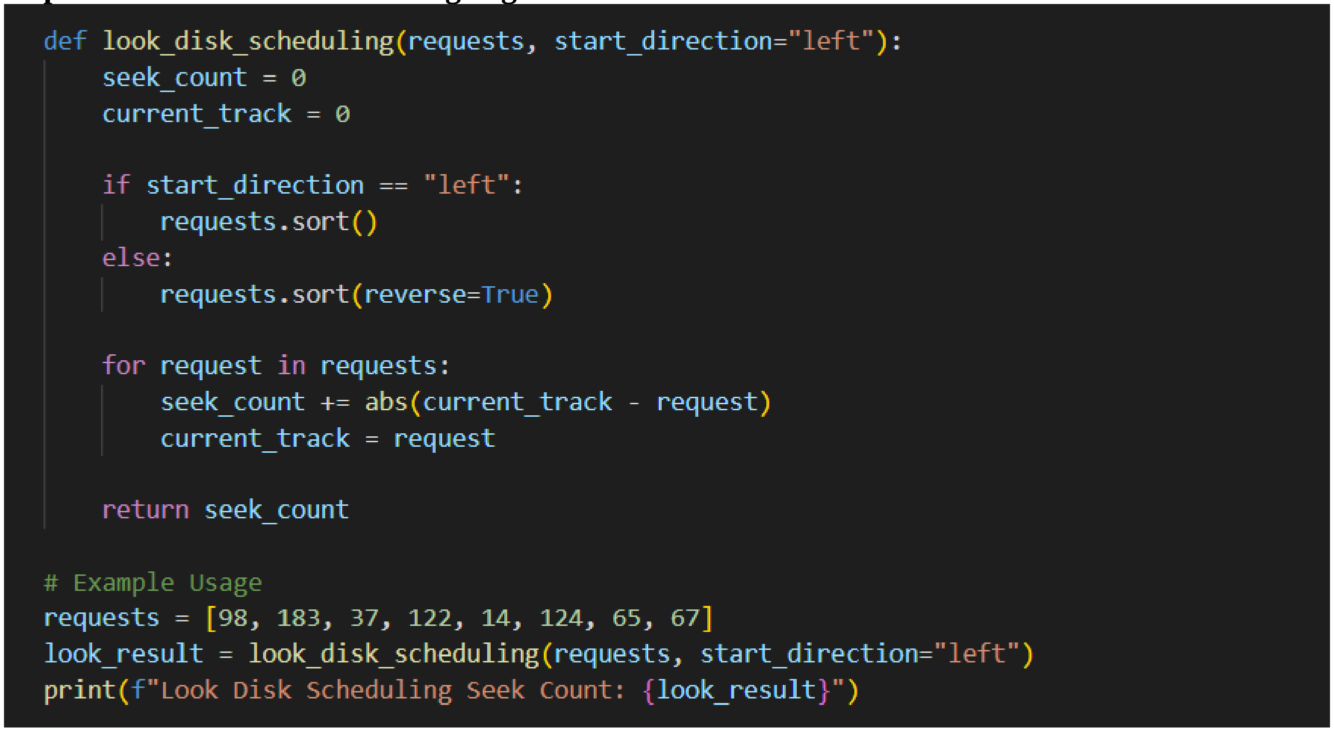
```python
def scan_disk_scheduling(requests, start_direction="left"):
    seek_count = 0
    current_track = 0

    if start_direction == "left":
        requests.sort()
    else:
        requests.sort(reverse=True)

    for request in requests:
        seek_count += abs(current_track - request)
        current_track = request

    return seek_count

# Example Usage
requests = [98, 183, 37, 122, 14, 124, 65, 67]
scan_result = scan_disk_scheduling(requests, start_direction="left")
print(f"SCAN Disk Scheduling Seek Count: {scan_result}")
```

45. Implement C-SCAN Disk Scheduling Algorithms.



```python
def c_scan_disk_scheduling(requests, start_direction="left"):
    seek_count = 0
    current_track = 0

    if start_direction == "left":
        requests.sort()
    else:
        requests.sort(reverse=True)

    for request in requests:
        seek_count += abs(current_track - request)
        current_track = request

    # Move to the other end of the disk
    seek_count += abs(current_track - (0 if start_direction == "left" else max(requests)))

    return seek_count

# Example Usage
requests = [98, 183, 37, 122, 14, 124, 65, 67]
c_scan_result = c_scan_disk_scheduling(requests, start_direction="left")
print(f"C-SCAN Disk Scheduling Seek Count: {c_scan_result}")
```

46. Implement Look Disk Scheduling Algorithms.



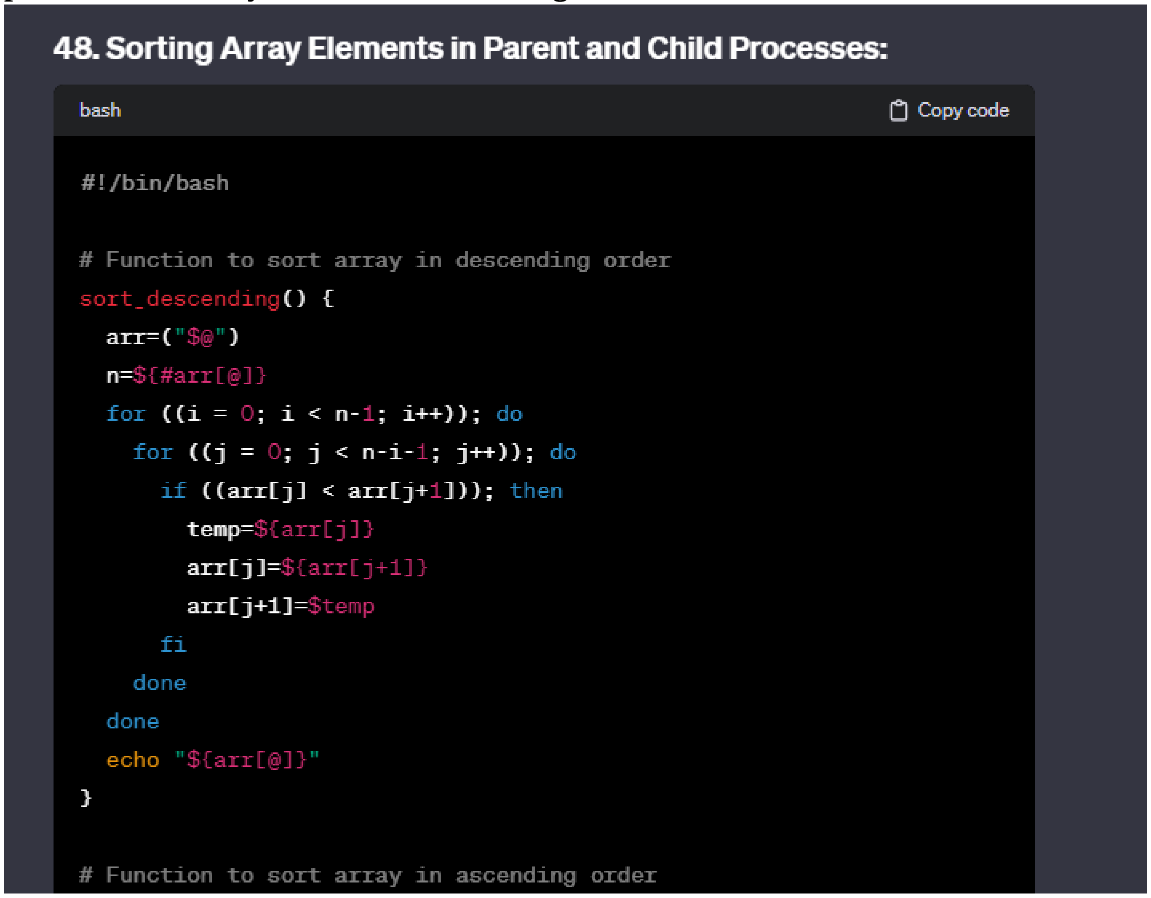
```python
def look_disk_scheduling(requests, start_direction="left"):
    seek_count = 0
    current_track = 0

    if start_direction == "left":
        requests.sort()
    else:
        requests.sort(reverse=True)

    for request in requests:
        seek_count += abs(current_track - request)
        current_track = request

    return seek_count

# Example Usage
requests = [98, 183, 37, 122, 14, 124, 65, 67]
look_result = look_disk_scheduling(requests, start_direction="left")
print(f"Look Disk Scheduling Seek Count: {look_result}")
```

47. Implement Look Disk Scheduling Algorithms.

Implement Multithreading to create child processes using fork() system call.

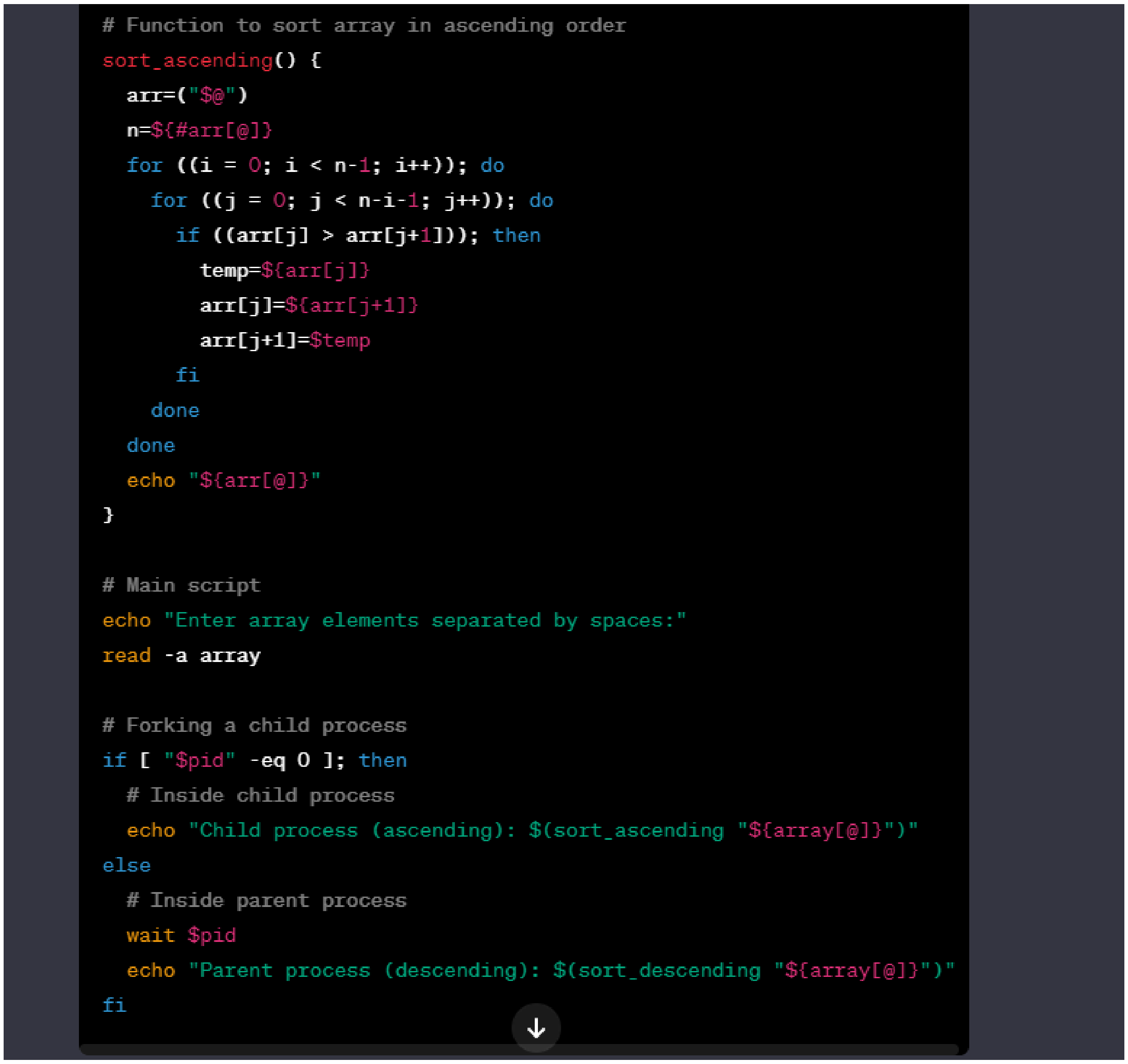
48. Program where parent process sorts array elements in descending order and child

process sorts array elements in ascending order.



48. Sorting Array Elements in Parent and Child Processes:

```bash
#!/bin/bash

# Function to sort array in descending order
sort_descending() {
  arr=("$@")
  n=${#arr[@]}
  for ((i = 0; i < n-1; i++)); do
    for ((j = 0; j < n-i-1; j++)); do
      if ((arr[j] < arr[j+1])); then
        temp=${arr[j]}
        arr[j]=${arr[j+1]}
        arr[j+1]=$temp
      fi
    done
  done
  echo "${arr[@]}"
}

# Function to sort array in ascending order
```

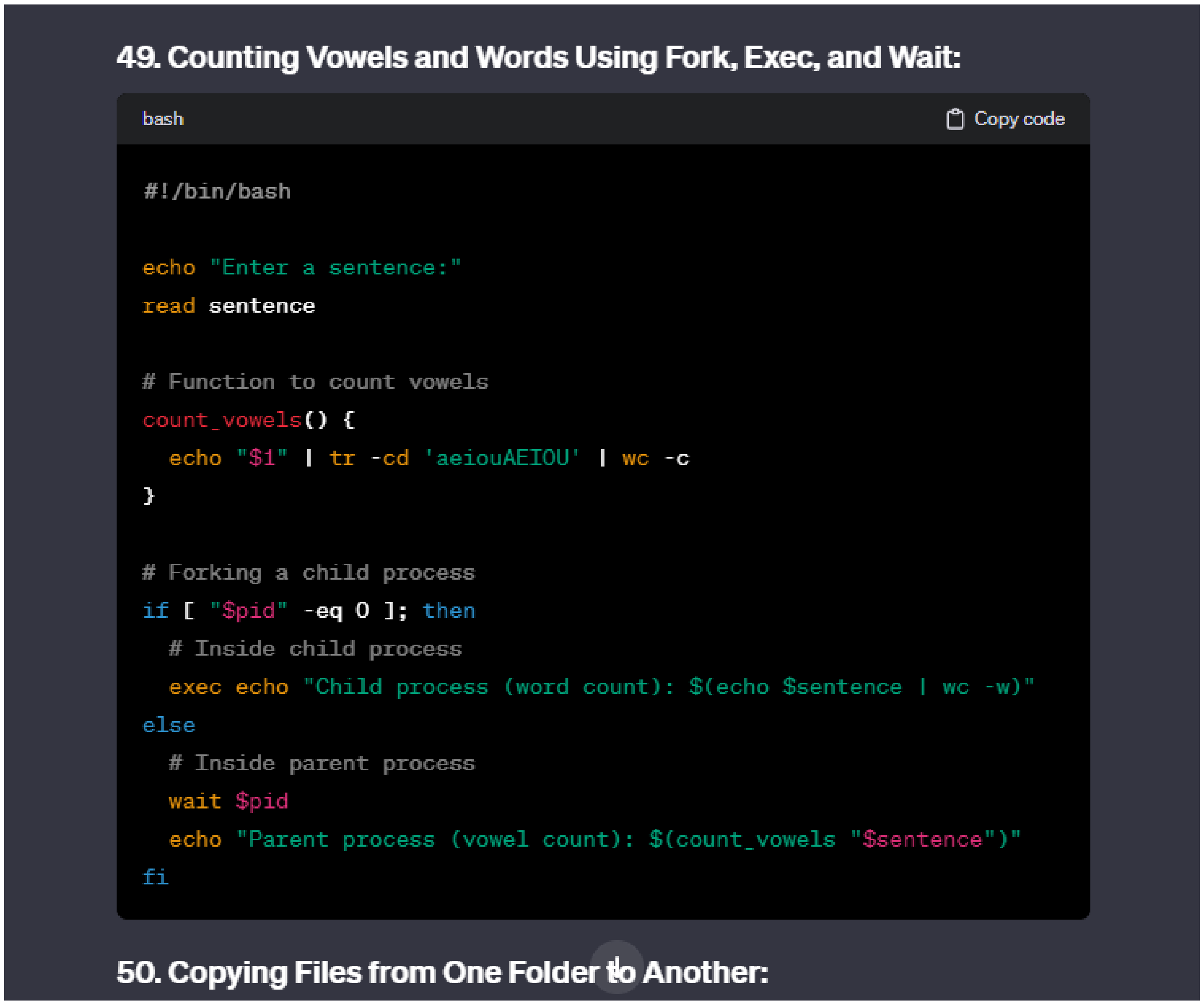


```
# Function to sort array in ascending order
sort_ascending() {
  arr=("$@")
  n=${#arr[@]}
  for ((i = 0; i < n-1; i++)); do
    for ((j = 0; j < n-i-1; j++)); do
      if ((arr[j] > arr[j+1])); then
        temp=${arr[j]}
        arr[j]=${arr[j+1]}
        arr[j+1]=$temp
      fi
    done
  done
  echo "${arr[@]}"
}

# Main script
echo "Enter array elements separated by spaces:"
read -a array

# Forking a child process
if [ "$pid" -eq 0 ]; then
  # Inside child process
  echo "Child process (ascending): $(sort_ascending "${array[@]}")"
else
  # Inside parent process
  wait $pid
  echo "Parent process (descending): $(sort_descending "${array[@]}")"
fi
```

49. Program where parent process Counts number of vowels in the given sentence and child process will count number of words in the same sentence. The above programs should use UNIX calls like fork, exec and wait. And also show the orphan and zombie states

50. Write Shell script to copy files from one folder to another

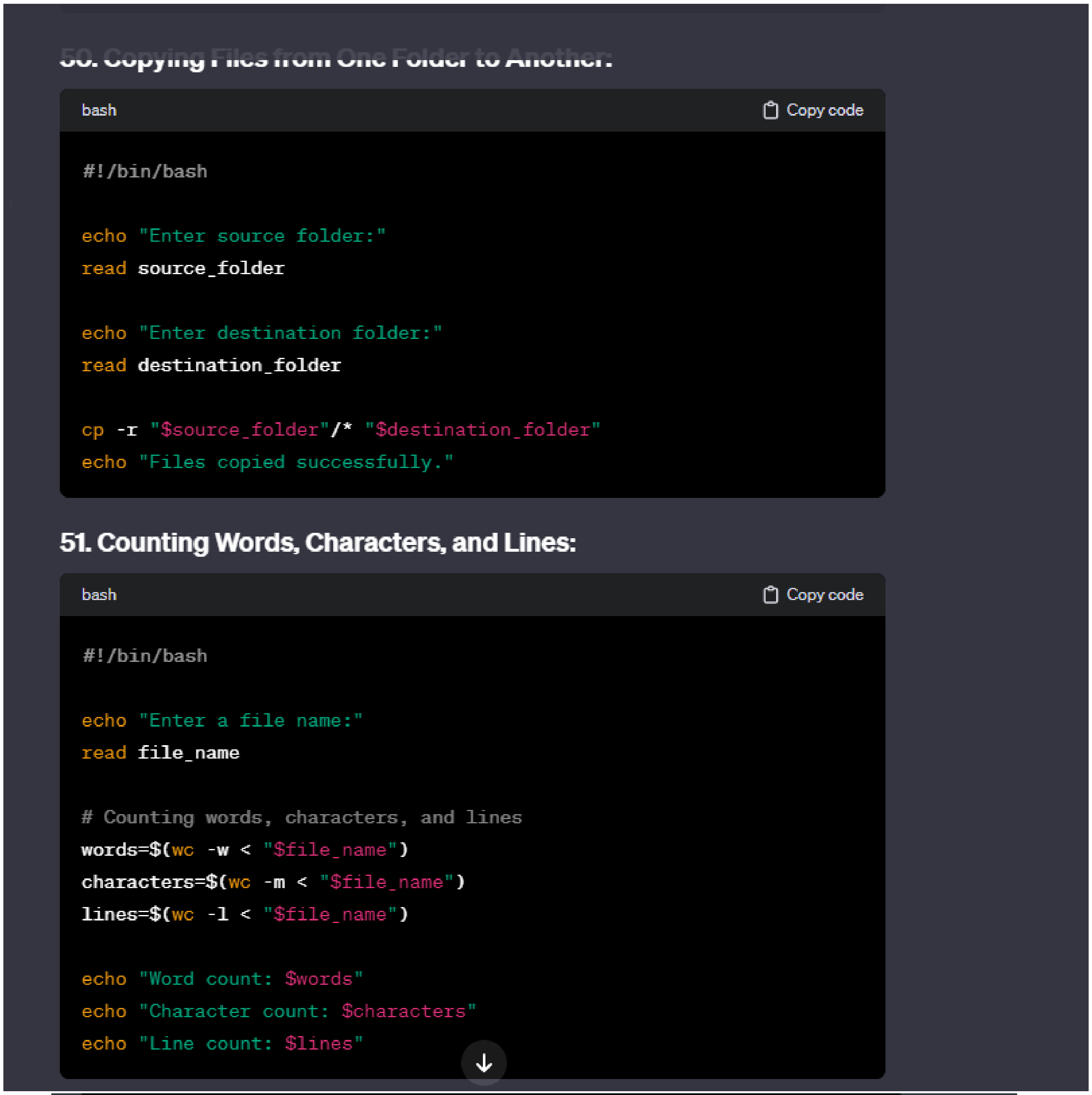
51. Write Shell script Count number of words, characters and lines.



49. Counting Vowels and Words Using Fork, Exec, and Wait:

```bash
#!/bin/bash

echo "Enter a sentence:"
read sentence

# Function to count vowels
count_vowels() {
  echo "$1" | tr -cd 'aeiouAEIOU' | wc -c
}

# Forking a child process
if [ "$pid" -eq 0 ]; then
  # Inside child process
  exec echo "Child process (word count): $(echo $sentence | wc -w)"
else
  # Inside parent process
  wait $pid
  echo "Parent process (vowel count): $(count_vowels "$sentence")"
fi
```

50. Copying Files from One Folder to Another:



50. Copying Files from One Folder to Another:

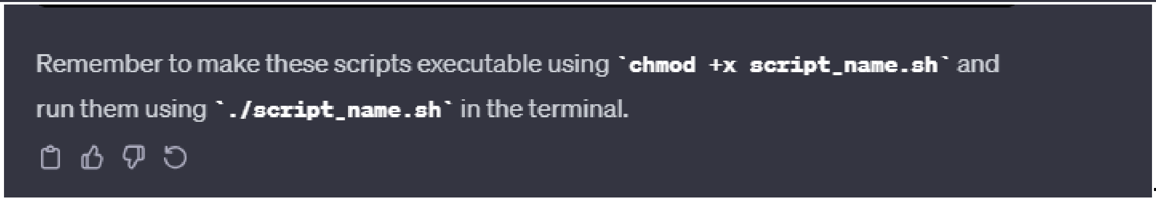
bash Copy code

```bash
#!/bin/bash

echo "Enter source folder:"
read source_folder

echo "Enter destination folder:"
read destination_folder

cp -r "$source_folder"/* "$destination_folder"
echo "Files copied successfully."
```

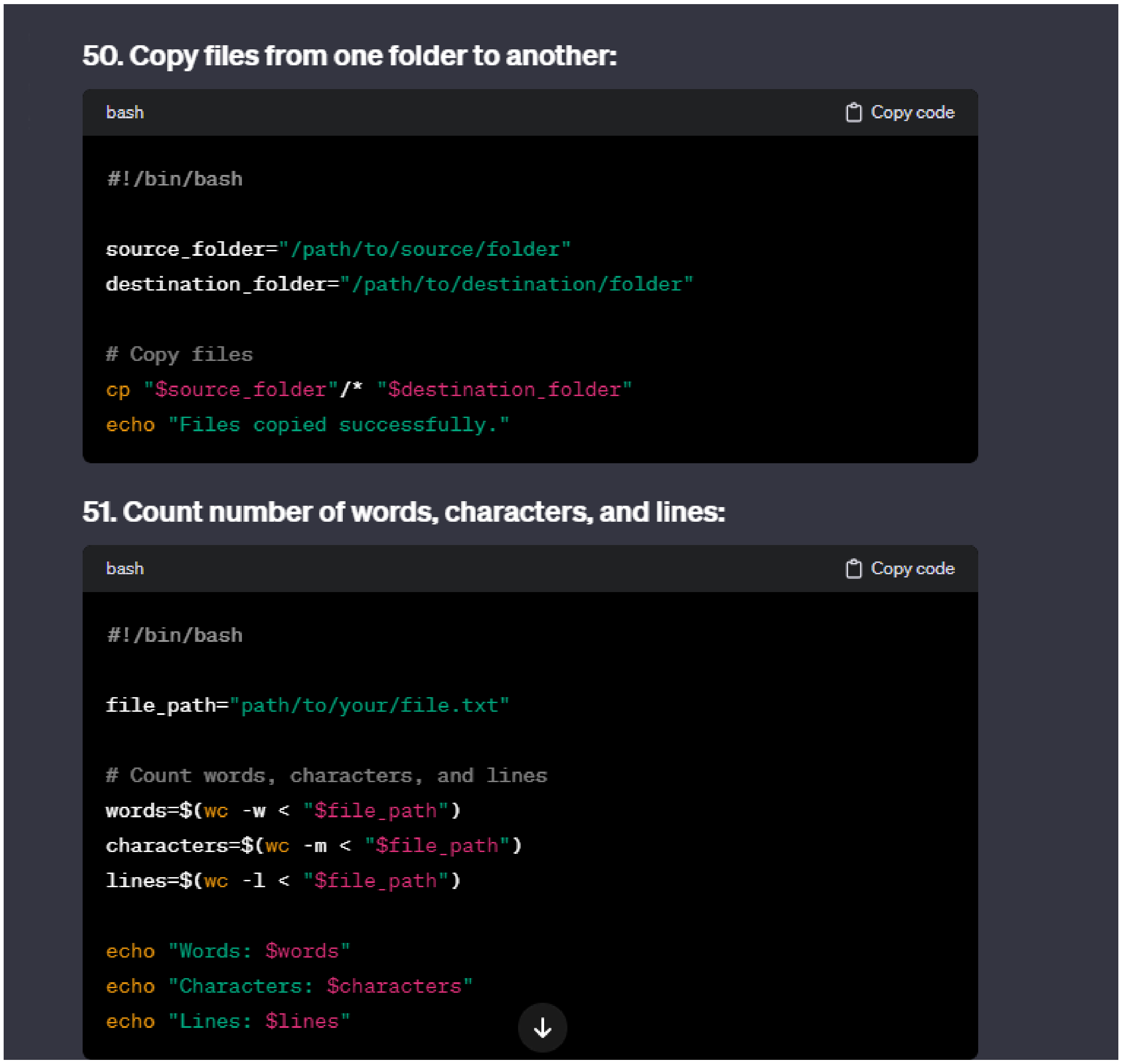
51. Counting Words, Characters, and Lines:

bash Copy code

```bash
#!/bin/bash

echo "Enter a file name:"
read file_name

# Counting words, characters, and lines
words=$(wc -w < "$file_name")
characters=$(wc -m < "$file_name")
lines=$(wc -l < "$file_name")

echo "Word count: $words"
echo "Character count: $characters"
echo "Line count: $lines"
```



Remember to make these scripts executable using `chmod +x script_name.sh` and run them using `./script_name.sh` in the terminal.



50. Copy files from one folder to another:

bash Copy code

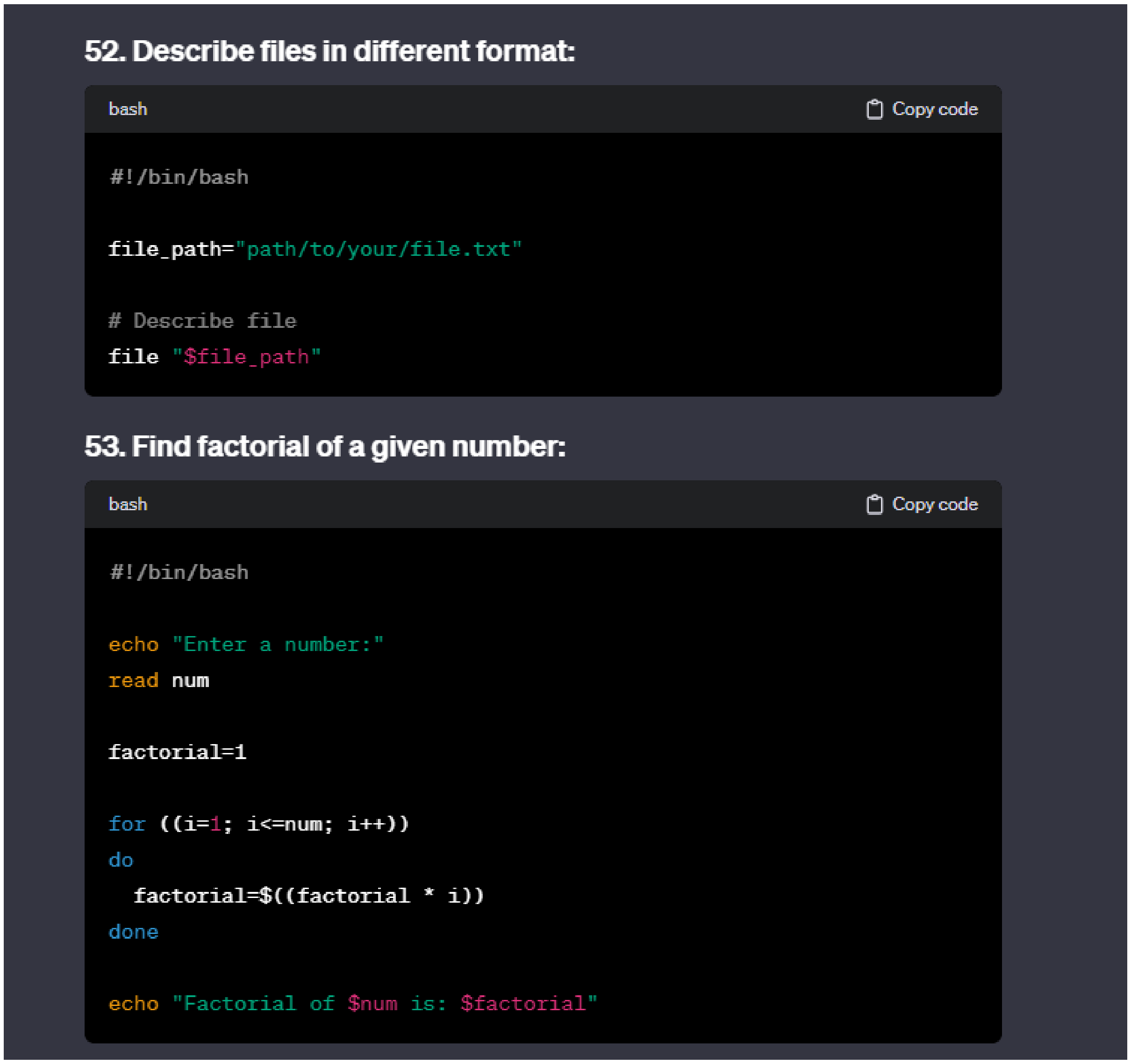
```bash
#!/bin/bash

source_folder="/path/to/source/folder"
destination_folder="/path/to/destination/folder"

# Copy files
cp "$source_folder"/* "$destination_folder"
echo "Files copied successfully."
```

51. Count number of words, characters, and lines:

bash Copy code

```bash
#!/bin/bash

file_path="path/to/your/file.txt"

# Count words, characters, and lines
words=$(wc -w < "$file_path")
characters=$(wc -m < "$file_path")
lines=$(wc -l < "$file_path")

echo "Words: $words"
echo "Characters: $characters"
echo "Lines: $lines"
```

52. Write Shell script To describe files in different format.
53. Write Shell script to find factorial of given number using bash script



52. Describe files in different format:

bash Copy code

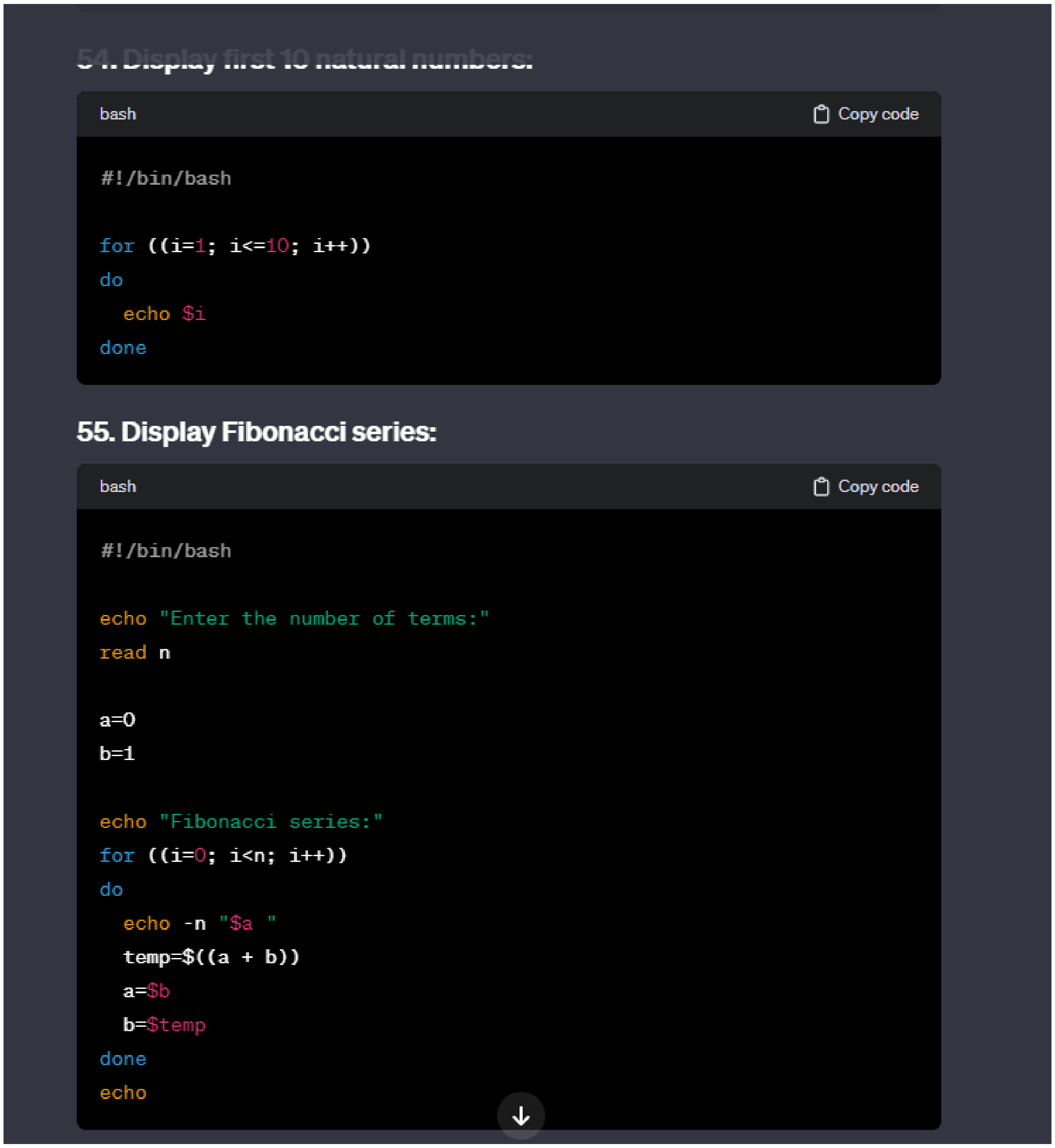
```bash
#!/bin/bash

file_path="path/to/your/file.txt"

# Describe file
file "$file_path"
```

53. Find factorial of a given number:

bash Copy code

```bash
#!/bin/bash

echo "Enter a number:"
read num

factorial=1

for ((i=1; i<=num; i++))
do
  factorial=$((factorial * i))
done

echo "Factorial of $num is: $factorial"
```

54. Display first 10 natural numbers using bash script
55. Display Fibonacci series using bash script


54. Display first 10 natural numbers:

bash Copy code

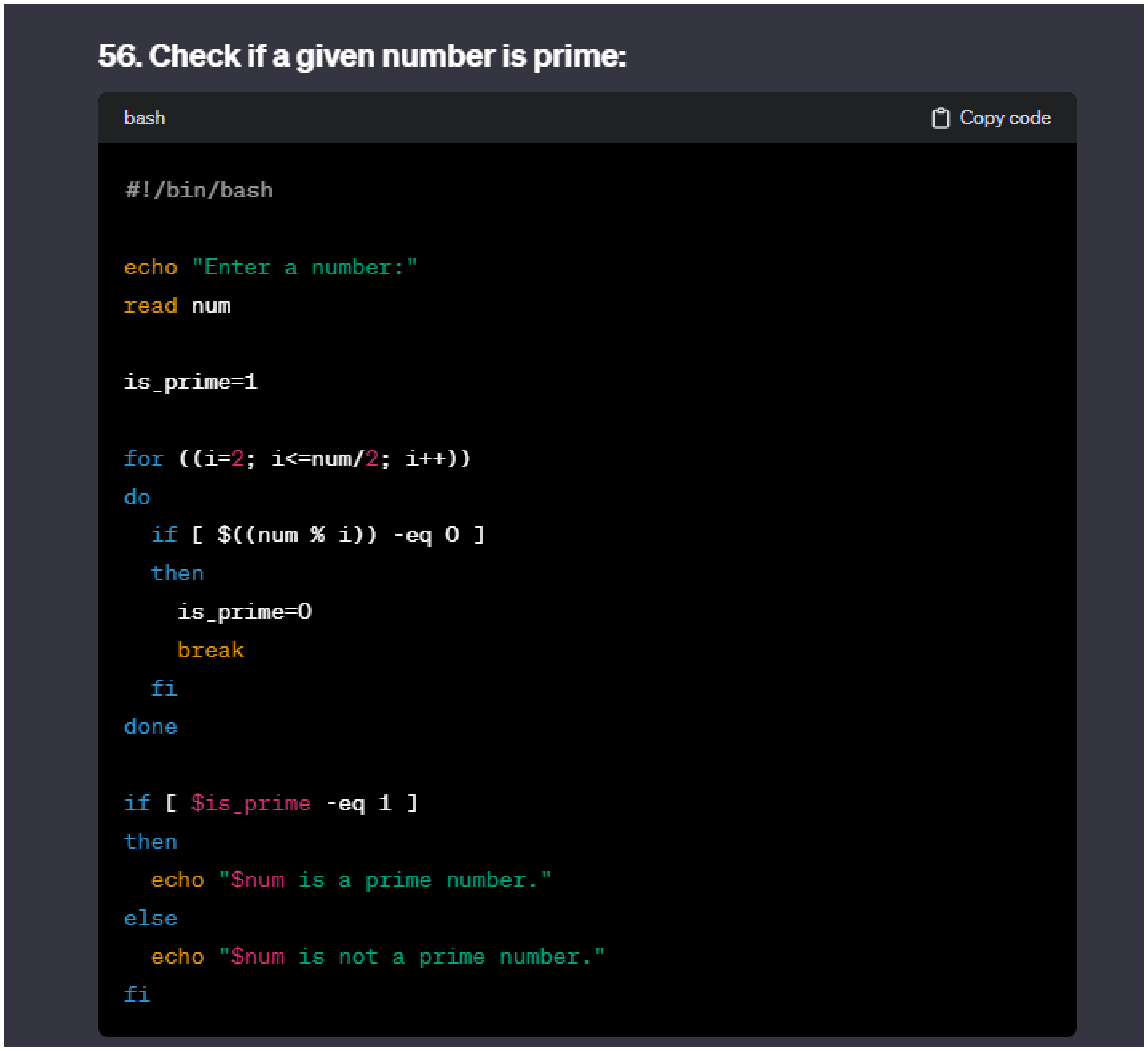
```bash
#!/bin/bash

for ((i=1; i<=10; i++))
do
  echo $i
done
```

55. Display Fibonacci series:

bash Copy code

```bash
#!/bin/bash

echo "Enter the number of terms:"
read n

a=0
b=1

echo "Fibonacci series:"
for ((i=0; i<n; i++))
do
  echo -n "$a "
  temp=$((a + b))
  a=$b
  b=$temp
done
echo
```

56. Find given number is prime or nor using bash script

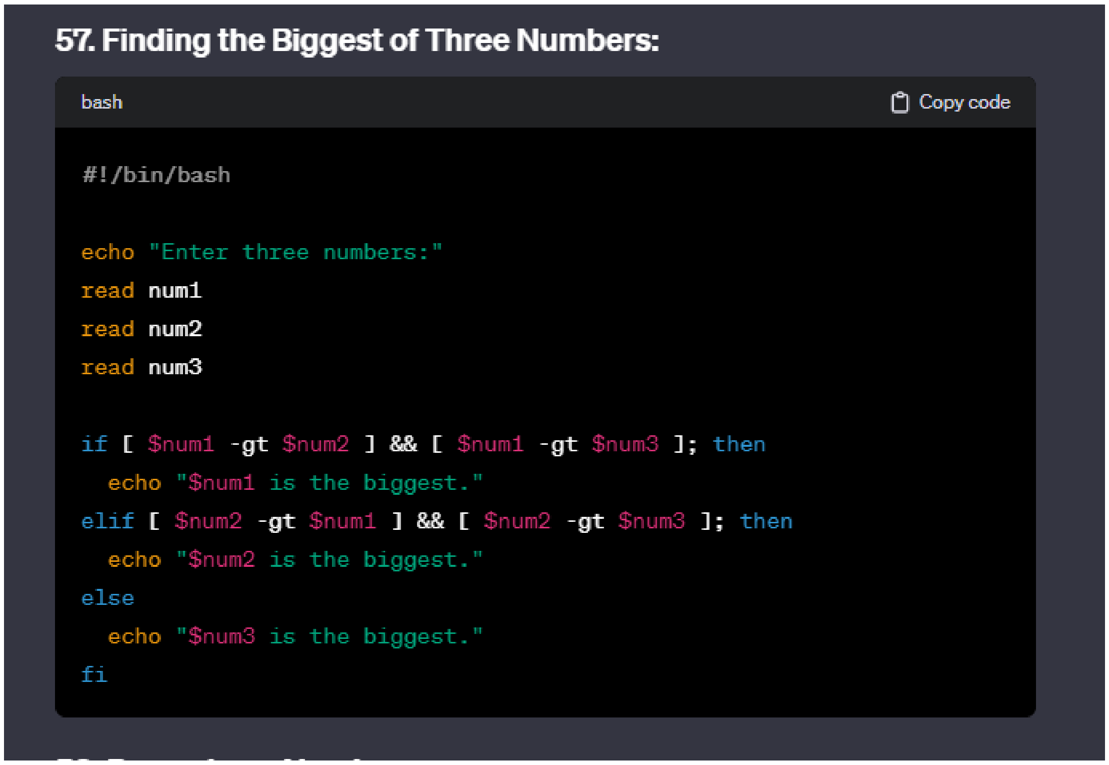


56. Check if a given number is prime:

bash Copy code

```bash
#!/bin/bash

echo "Enter a number:"
read num

is_prime=1

for ((i=2; i<=num/2; i++))
do
  if [ $((num % i)) -eq 0 ]
  then
    is_prime=0
    break
  fi
done

if [ $is_prime -eq 1 ]
then
  echo "$num is a prime number."
else
  echo "$num is not a prime number."
fi
```

57. **Write shell script to finding biggest of three numbers**

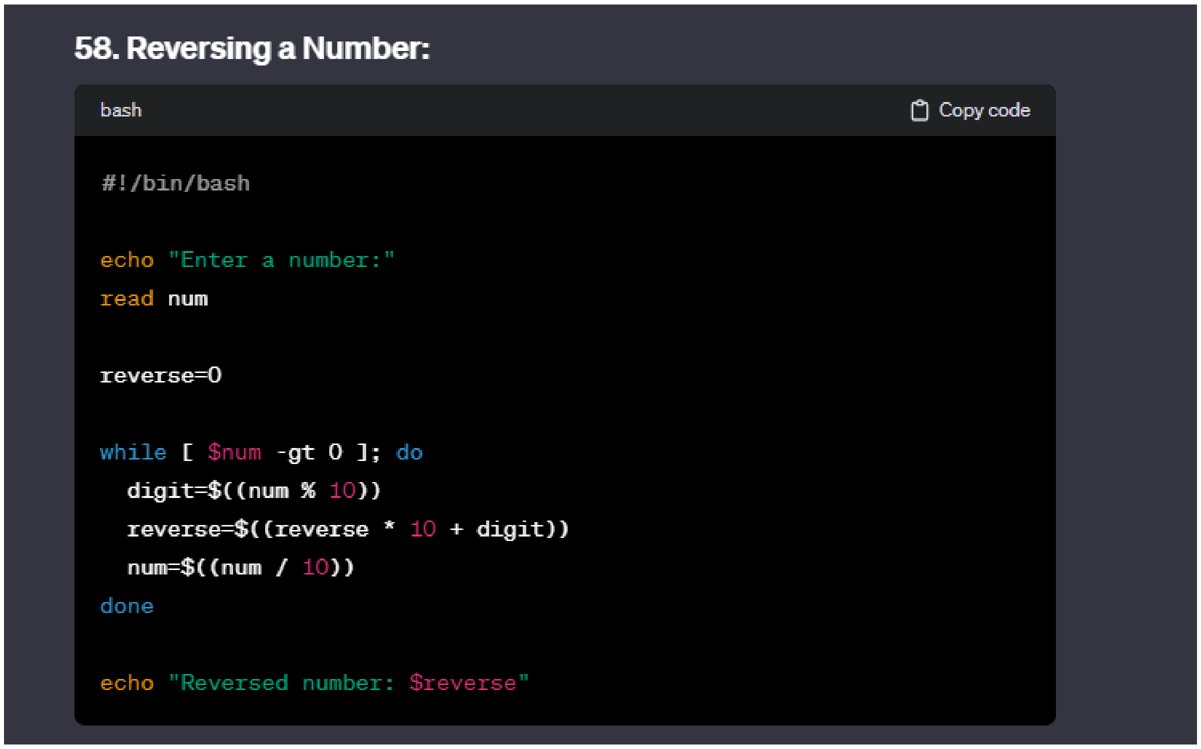


57. Finding the Biggest of Three Numbers:

bash Copy code

```bash
#!/bin/bash

echo "Enter three numbers:"
read num1
read num2
read num3

if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]; then
  echo "$num1 is the biggest."
elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]; then
  echo "$num2 is the biggest."
else
  echo "$num3 is the biggest."
fi
```

58. Write shell script to reversing a number

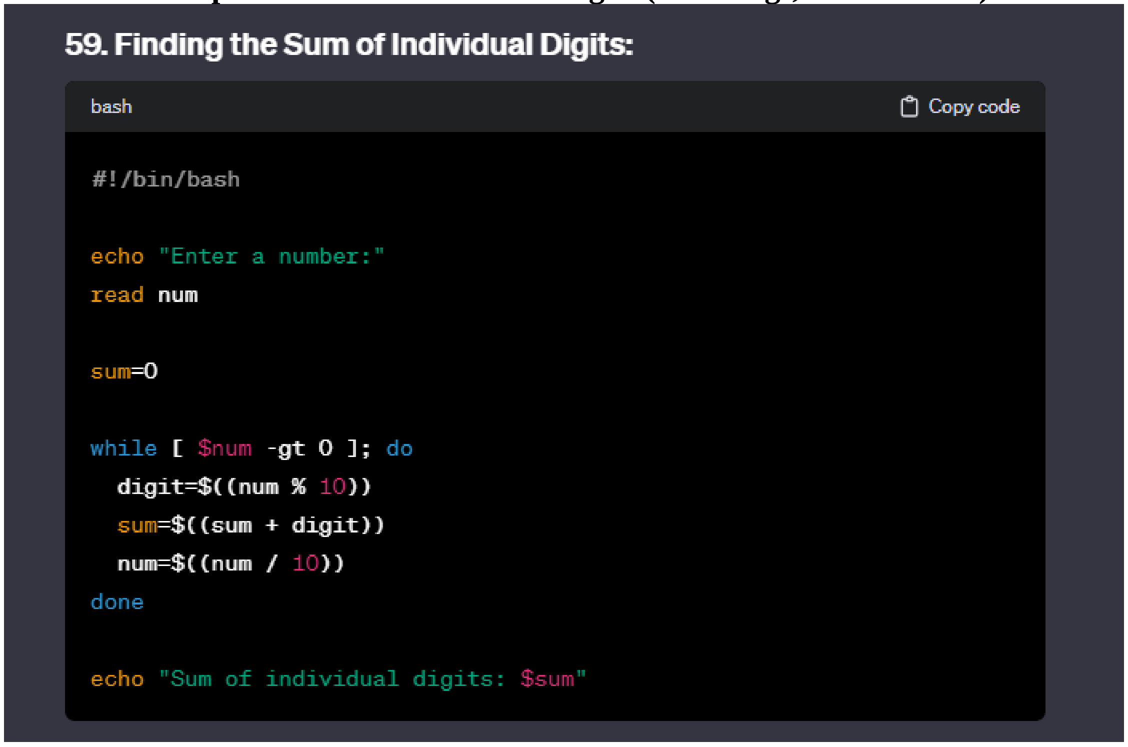


58. Reversing a Number:

bash Copy code

```bash
#!/bin/bash

echo "Enter a number:"
read num

reverse=0

while [ $num -gt 0 ]; do
  digit=$((num % 10))
  reverse=$((reverse * 10 + digit))
  num=$((num / 10))
done

echo "Reversed number: $reverse"
```

59. Write shell script find Sum of individual digits (1234 -&gt; 1+2+3+4=10)



59. Finding the Sum of Individual Digits:

bash Copy code

```bash
#!/bin/bash

echo "Enter a number:"
read num

sum=0

while [ $num -gt 0 ]; do
  digit=$((num % 10))
  sum=$((sum + digit))
  num=$((num / 10))
done

echo "Sum of individual digits: $sum"
```