Republic of the Philippines
CAMARINES SUR POLYTECHNIC COLLEGES
Nabua, Camarines Sur

COLLEGE *of* COMPUTER STUDIES

CCCS 104 - Data Structures and Algorithms

LEARNING TASK (LINEAR DATA STRUCTURE - QUEUE)

GROUP NO: 9 SECTION: BSCS 2A

GROUP LEADER: Venn P. Delos Santos

GROUP MEMBERS: Marc Christian D. Tumaneng

John Mark A. Pajenago

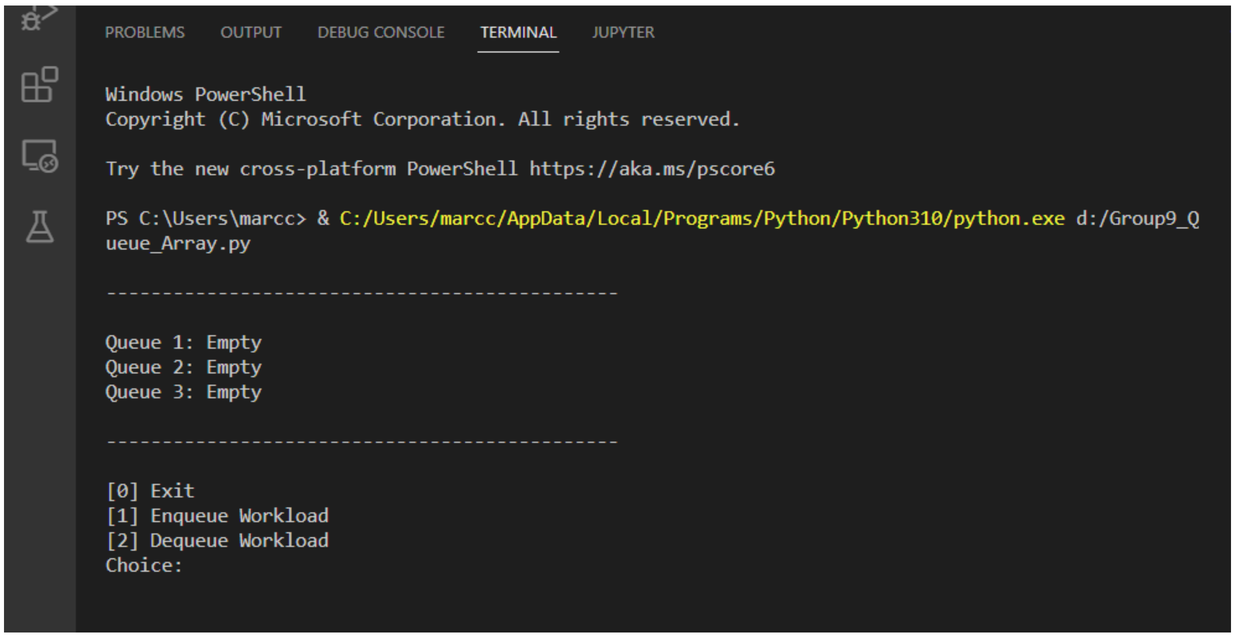
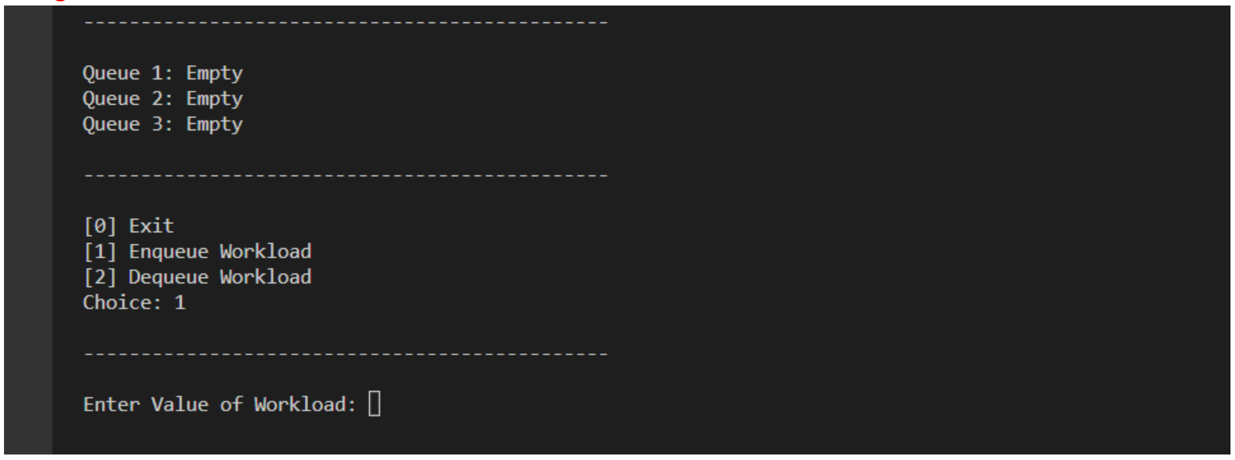
RATIONALE

A Queue is an important linear data structure in programming. A Queue follows the FIFO (First In First Out) method and is open at both of its ends. Data insertion is done at one end, rear end or the tail of the queue while deletion is done at the other end called the front end or the head of the queue. We define a queue to be a list in which all additions to the list are made at one end, and all deletions from the list are made at the other end. The element which is first pushed into the order, the operation is first performed on that. Some real life examples include a line of people waiting to buy a ticket at a cinema hall, a cashier line in a supermarket, people on an escalator and so on. Queues are typically used to manage threads in multithreading and implementing priority queuing systems. As a queue data structure is an ordered list and yet open at both ends, it has multifold applications in the real world such as Job or CPU Scheduling, Traffic System, Routers and switches in networking, Maintaining the playlist in media players and so on.

The Python program that we created uses exactly this Queue Linear Data Structure with the help and implementation of array. The program asks the user to select from the choices on what kind of operation to perform such as Exit, Enqueue, or Dequeue Workload. When the user enters the value of the workload that is asked by the program, it does the enqueue operation to the queue with the least total sum of workload. It also executes the dequeue operation to the queue with the lowest head or first element. Finally, the program prints the results in every operation and continues to loop until the user selects the choice in exiting the program.

USER GUIDE

Step by step instructions on how to use your program. Include images for easily visualization

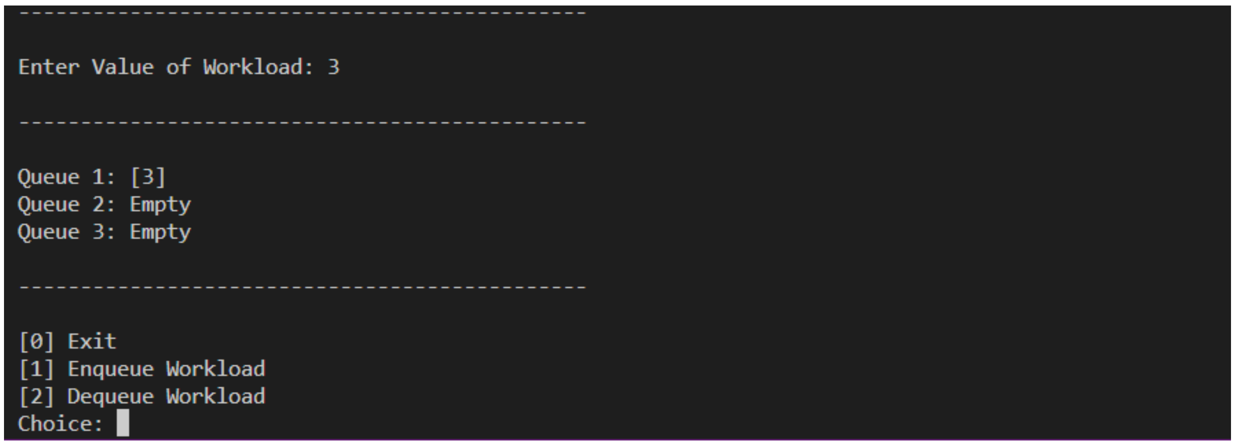
Step1 ***Run the program***
Image 1 *The program will ask for the user's choice: [0] - Exit, [1] - Enqueue workload, [2] -Dequeue workload, the program won't stop/end until the user decides to select [0].* 
Step 2 *Since the queue is empty we have to* ***enqueue a number, select choice 1****.*
Image 2 

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\marcc> & C:/Users/marcc/AppData/Local/Programs/Python/Python310/python.exe d:/Group9_Q
ueue_Array.py

---------------------------------------------

Queue 1: Empty
Queue 2: Empty
Queue 3: Empty

---------------------------------------------

[0] Exit
[1] Enqueue Workload
[2] Dequeue Workload
Choice:
```

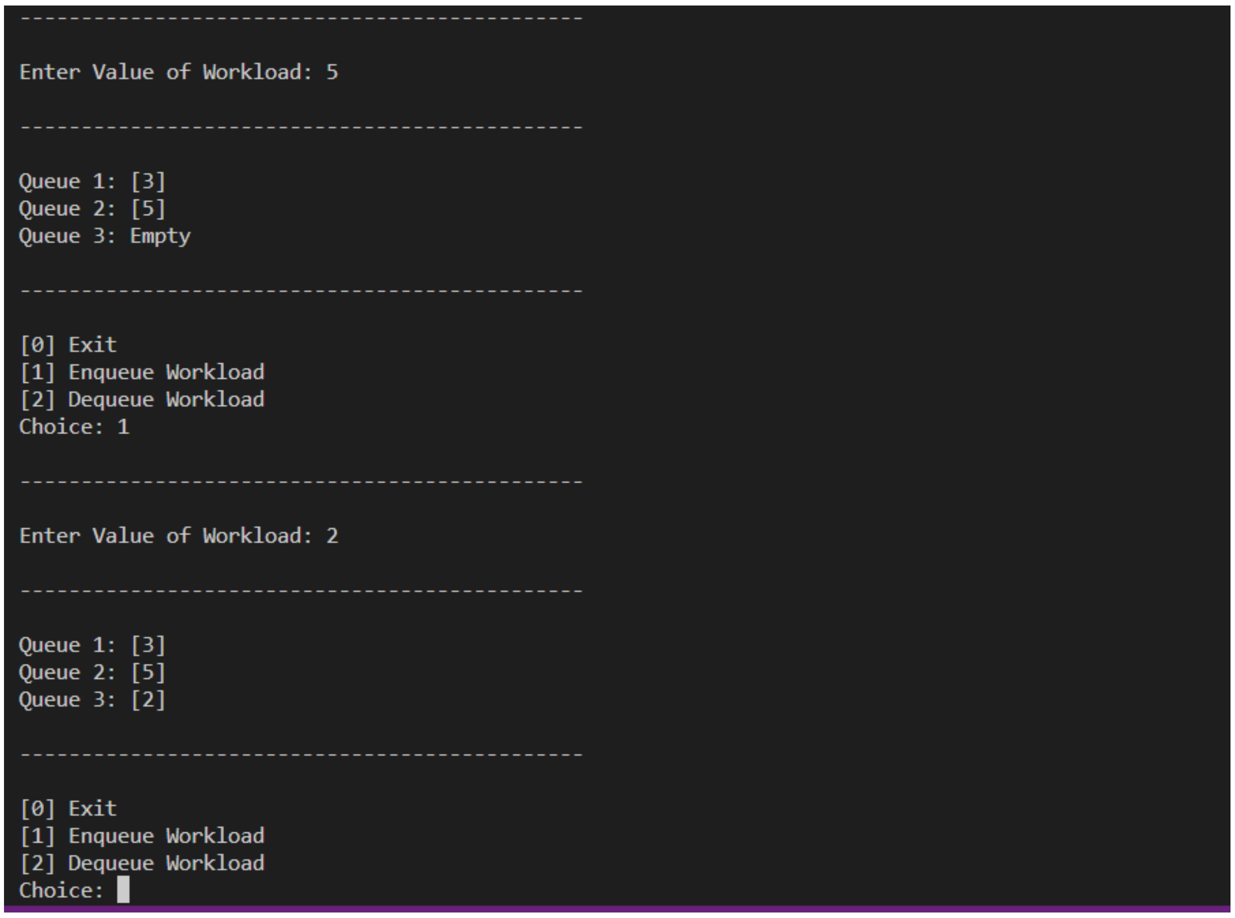
```
---------------------------------------------

Queue 1: Empty
Queue 2: Empty
Queue 3: Empty

---------------------------------------------

[0] Exit
[1] Enqueue Workload
[2] Dequeue Workload
Choice: 1

---------------------------------------------

Enter Value of Workload:
```

Step 3
After selecting choice 1, the program will ask the user to enter the value of workload to enqueue. ***Enter the integer value of workload.***

Image 3
The program will enqueue the number of workload that the user input to the queue with the least total of workload:



```
----------------------------------------------

Enter Value of Workload: 3

----------------------------------------------

Queue 1: [3]
Queue 2: Empty
Queue 3: Empty

----------------------------------------------

[0] Exit
[1] Enqueue Workload
[2] Dequeue Workload
Choice: 
```

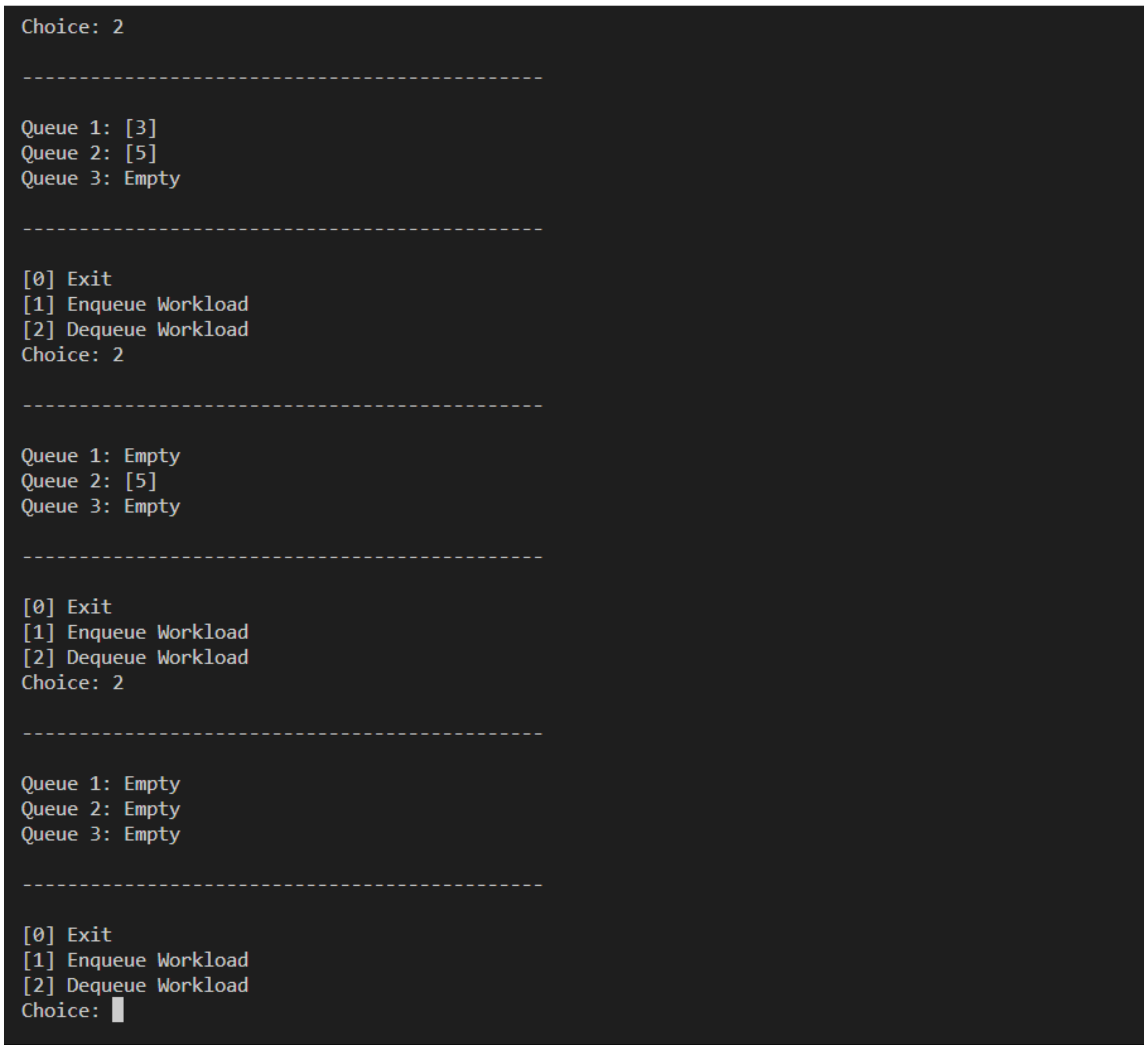


```
----------------------------------------------

Enter Value of Workload: 5

----------------------------------------------

Queue 1: [3]
Queue 2: [5]
Queue 3: Empty

----------------------------------------------

[0] Exit
[1] Enqueue Workload
[2] Dequeue Workload
Choice: 1

----------------------------------------------

Enter Value of Workload: 2

----------------------------------------------

Queue 1: [3]
Queue 2: [5]
Queue 3: [2]

----------------------------------------------

[0] Exit
[1] Enqueue Workload
[2] Dequeue Workload
Choice: 
```

Step 4
Now that we are done with the enqueue operation and we now have some numbers/ elements to our queue, next is we ***Dequeue workload by choosing Choice 2 .***

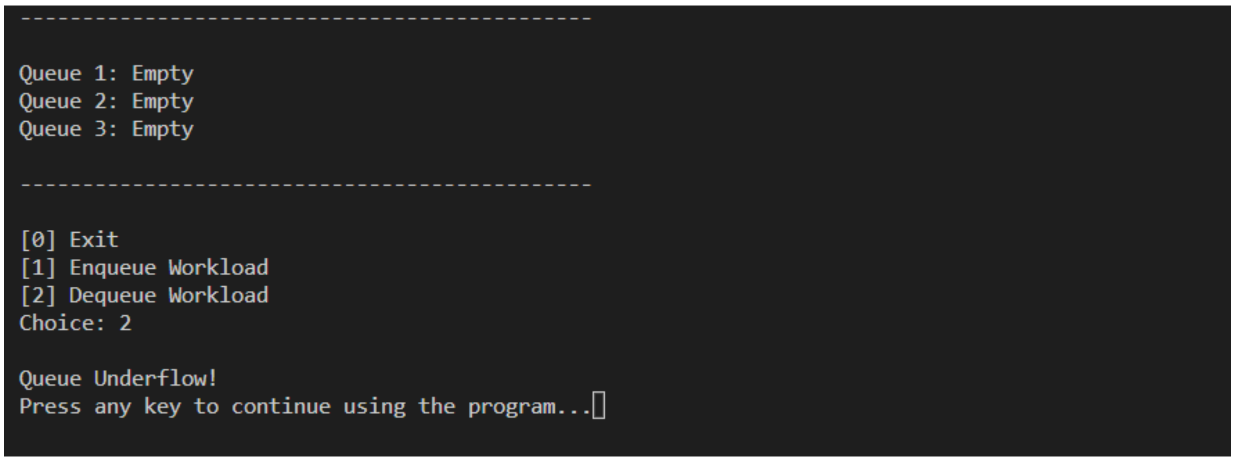
Image 4

The program will dequeue the first element of the queue with the lowest head value among all of the queues.



```
Choice: 2

----------------------------------------------

Queue 1: [3]
Queue 2: [5]
Queue 3: Empty

----------------------------------------------

[0] Exit
[1] Enqueue Workload
[2] Dequeue Workload
Choice: 2

----------------------------------------------

Queue 1: Empty
Queue 2: [5]
Queue 3: Empty

----------------------------------------------

[0] Exit
[1] Enqueue Workload
[2] Dequeue Workload
Choice: 2

----------------------------------------------

Queue 1: Empty
Queue 2: Empty
Queue 3: Empty

----------------------------------------------

[0] Exit
[1] Enqueue Workload
[2] Dequeue Workload
Choice: 
```

if we perform dequeue to the queues that are already empty, the program will display "Queue Underflow!"

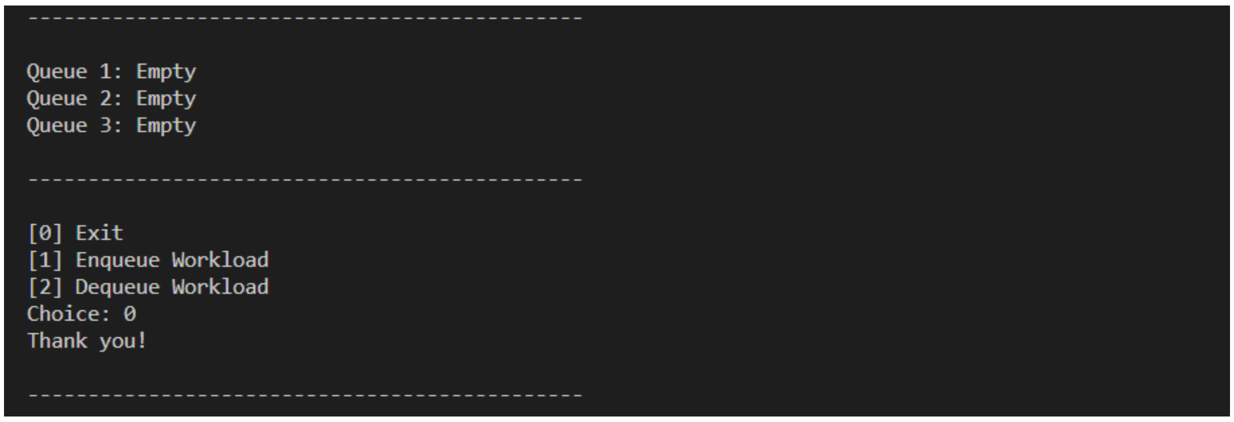


```
----------------------------------------------

Queue 1: Empty
Queue 2: Empty
Queue 3: Empty

----------------------------------------------

[0] Exit
[1] Enqueue Workload
[2] Dequeue Workload
Choice: 2

Queue Underflow!
Press any key to continue using the program...
```

Step 5
to end the program ***select [0] to exit***

Image 5

The program will display "Thank You!" and terminate the program.



```
--------------------------------------------

Queue 1: Empty
Queue 2: Empty
Queue 3: Empty

--------------------------------------------

[0] Exit
[1] Enqueue Workload
[2] Dequeue Workload
Choice: 0
Thank you!

--------------------------------------------
```

PROGRAM CODE

```python
# GROUP 9
# Queue implementation using Array in Python

class Queue:

    def __init__(self):
        self.queue = []
        self.sum = 0

    # Add an element
    def enqueue(self, item):
        self.queue.append(item)
        self.sum += item

     # Remove an element
    def dequeue(self):
        if len(self.queue) < 1:
            return None
        self.sum -= self.queue[0]
        return self.queue.pop(0)

    # Display an element
    def display(self):
        print(self.queue)

    # Get the size of the queue
    def size(self):
        return len(self.queue)

    # Head pointer
    def head(self):
        return self.queue[0]

# Display all the queues and their elements
def printQueue(queue, queue2, queue3):
    print("Queue 1: ", end='')
    queue.display() if queue.size() != 0 else print("Empty")
    print("Queue 2: ", end='')
    queue2.display() if queue2.size() != 0 else print("Empty")
    print("Queue 3: ", end='')
    queue3.display() if queue3.size() != 0 else print("Empty")
    print('\n----------------------------------------------''\n')

print('\n----------------------------------------------''\n')

# 3 queues implemented using an array
queue = Queue()
queue2 = Queue()
queue3 = Queue()
printQueue(queue, queue2, queue3)

# loop for starting the program again
while True:
    # menu
    print('[0] Exit')
    print('[1] Enqueue Workload')
    print('[2] Dequeue Workload')
    choice = int(input("Choice: "))

    # if the user selects 0, the program will end
    if choice == 0:
        print("Thank you!")
        print('\n----------------------------------------------''\n')
        break

    # if the user selects 1, the enqueue operation will be done to the queue
with the least total work load
    elif choice == 1:
        print('\n----------------------------------------------''\n')
        workVal = int(input("Enter Value of Workload: "))
        if queue.sum <= queue2.sum and queue.sum <= queue3.sum:
            queue.enqueue(workVal)
        elif queue2.sum <= queue.sum and queue2.sum <= queue3.sum:
            queue2.enqueue(workVal)
        elif queue3.sum <= queue.sum and queue3.sum <= queue2.sum:
            queue3.enqueue(workVal)
        else:
            queue.enqueue(workVal)

    # if the user selects 2, the dequeue operation will be done to the queue
with the lowest first element
    elif choice == 2:
        # Error message when trying to dequeue from an empty queue
        if queue.size() == 0 and queue2.size() == 0 and queue3.size() == 0:
            print("\nQueue Underflow!")
            input("Press any key to continue using the program...")
            print('\n----------------------------------------------''\n')

        # conditional statements to determine the queue in which dequeue
operation will be done
        elif queue.size() == 0:
            if queue3.size() == 0:
                queue2.dequeue()
                print('\n----------------------------------------------''\n')
                printQueue(queue, queue2, queue3)
                continue
            if queue2.size() == 0:
                queue3.dequeue()
                print('\n----------------------------------------------''\n')
                printQueue(queue, queue2, queue3)
                continue
            if queue2.head() <= queue3.head():
                queue2.dequeue()
            else:
                queue3.dequeue()

        elif queue2.size() == 0:
            if queue3.size() == 0:
                queue.dequeue()
                print('\n----------------------------------------------''\n')
                printQueue(queue, queue2, queue3)
                continue
            if queue.head() <= queue3.head():
                queue.dequeue()
            else:
                queue3.dequeue()

        elif queue3.size() == 0:
            if queue.head() <= queue2.head():
                queue.dequeue()
            else:
                queue2.dequeue()

        elif queue.head() <= queue2.head() and queue.head() <= queue3.head():
            queue.dequeue()
        elif queue2.head() <= queue.head() and queue2.head() <= queue3.head():
            queue2.dequeue()
        elif queue3.head() <= queue2.head() and queue3.head() <= queue.head():
            queue3.dequeue()
        else:
            queue.dequeue()

    print('\n----------------------------------------------''\n')
    # function call to print all the queues and their elements
    printQueue(queue, queue2, queue3)
```

TUTORIAL VIDEO

YouTube Link: https://youtu.be/FLJnDL4cCQs

TAKEAWAYS

Name of Member:Venn P. Delos Santos
Contribution to the Group: Coding the program, recording, documentation
Learnings: Understand the concept of Queue Linear Data Structure and on how to perform operations like enqueue and dequeue in an array implementation.

Name of Member:Marc Christian D. Tumaneng
Contribution to the Group: Coding the program, recording, documentation
Learnings: I understand the concept and implementation of Queue Abstract Data Structure using Array.

Name of Member:John Mark A. Pajenago
Contribution to the Group: Coding the program, recording, documentation
Learnings:I learned that how to know the use of queue and what are the function that use in queue.