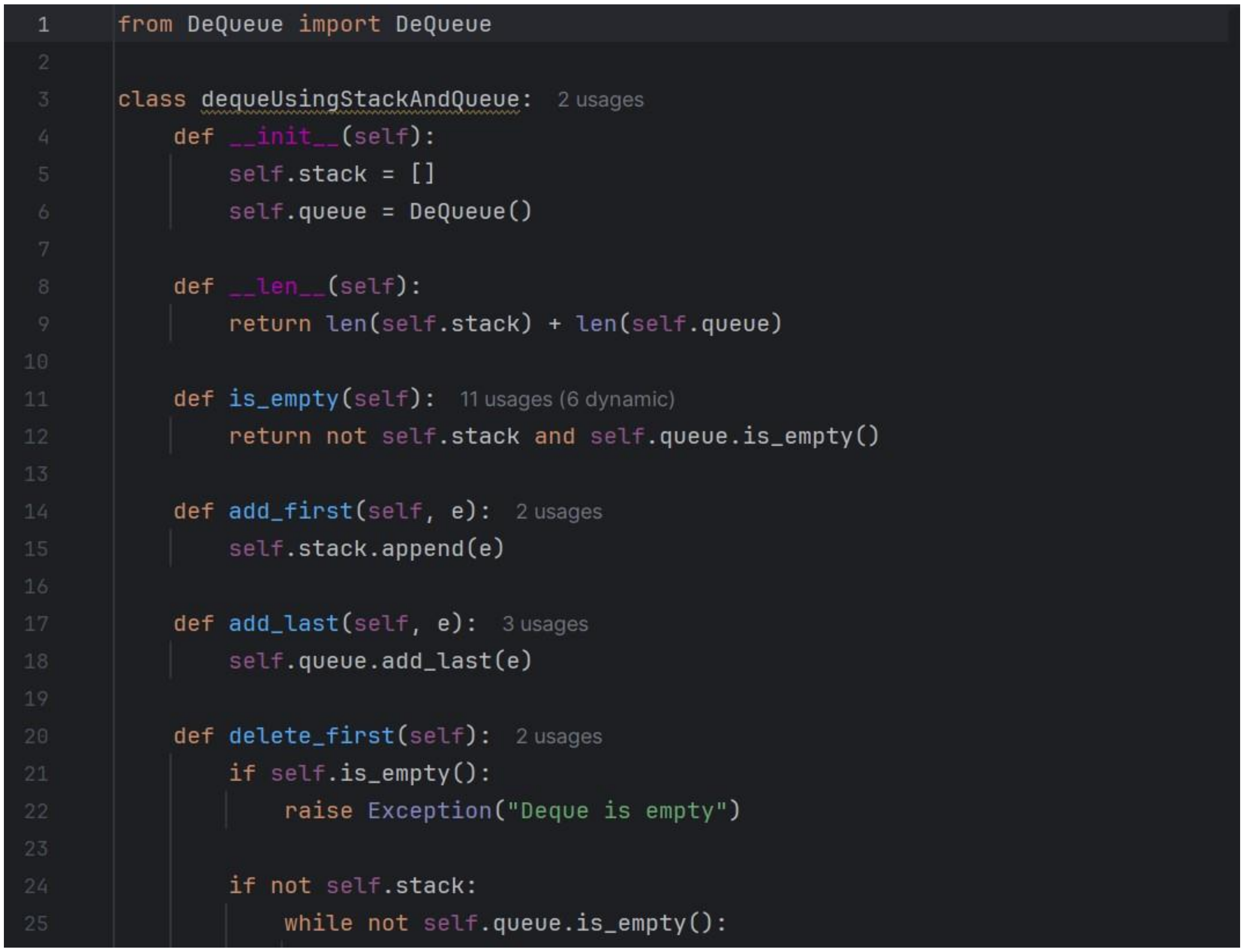
UNIVERSITY OF Baguio

SCHOOL OF INFORMATION TECHNOLOGY

NAME: MARK LL OYD YADAO /RICHMOND BROQUEZA		
DSALGO1/IDB2	DATE SUBMITTED:12/062024	

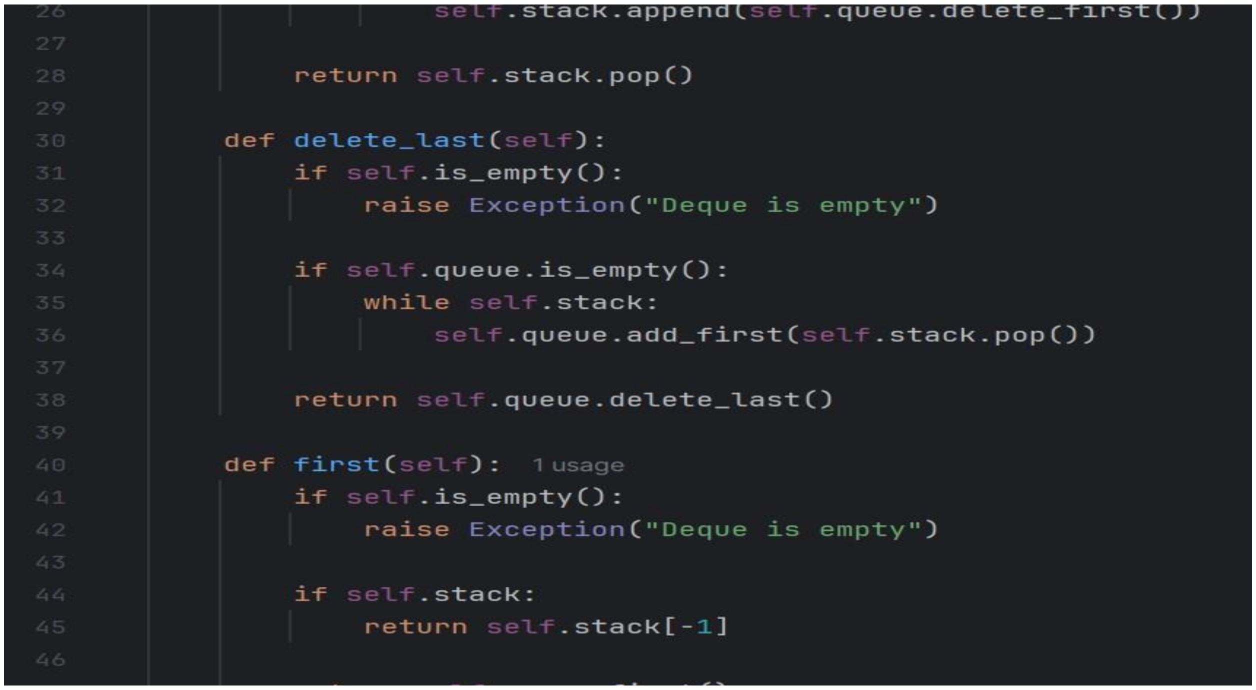
TERM PROJECT 1



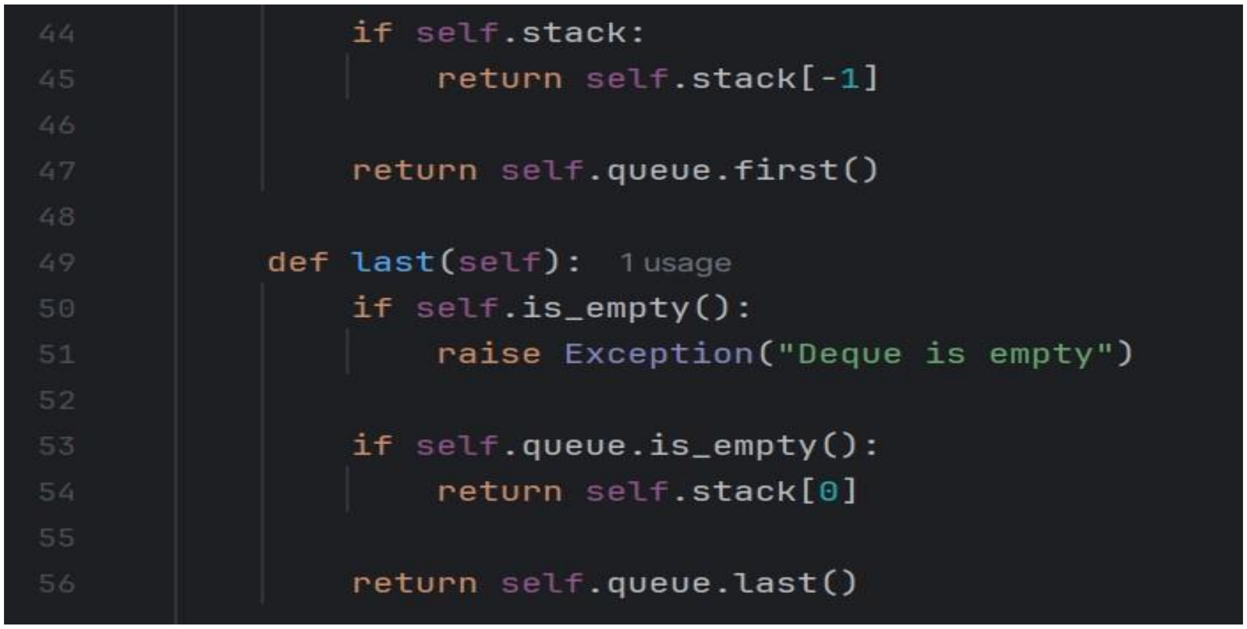
```
from DeQueue import DeQueue

class dequeUsingStackAndQueue:  2 usages
    def __init__(self):
        self.stack = []
        self.queue = DeQueue()

    def __len__(self):
        return len(self.stack) + len(self.queue)

    def is_empty(self):  11 usages (6 dynamic)
        return not self.stack and self.queue.is_empty()

    def add_first(self, e):  2 usages
        self.stack.append(e)

    def add_last(self, e):  3 usages
        self.queue.add_last(e)

    def delete_first(self):  2 usages
        if self.is_empty():
            raise Exception("Deque is empty")

        if not self.stack:
            while not self.queue.is_empty():
```

SCHOOL OF INFORMATION TECHNOLOGY

NAME: MARK LL OYD YADAO /RICHMOND BROQUEZA		
DSALGO1/IDB2	DATE SUBMITTED:12/062024	



```
                self.stack.append(self.queue.delete_first())

        return self.stack.pop()

    def delete_last(self):
        if self.is_empty():
            raise Exception("Deque is empty")

        if self.queue.is_empty():
            while self.stack:
                self.queue.add_first(self.stack.pop())

        return self.queue.delete_last()

    def first(self):  1 usage
        if self.is_empty():
            raise Exception("Deque is empty")

        if self.stack:
            return self.stack[-1]

```

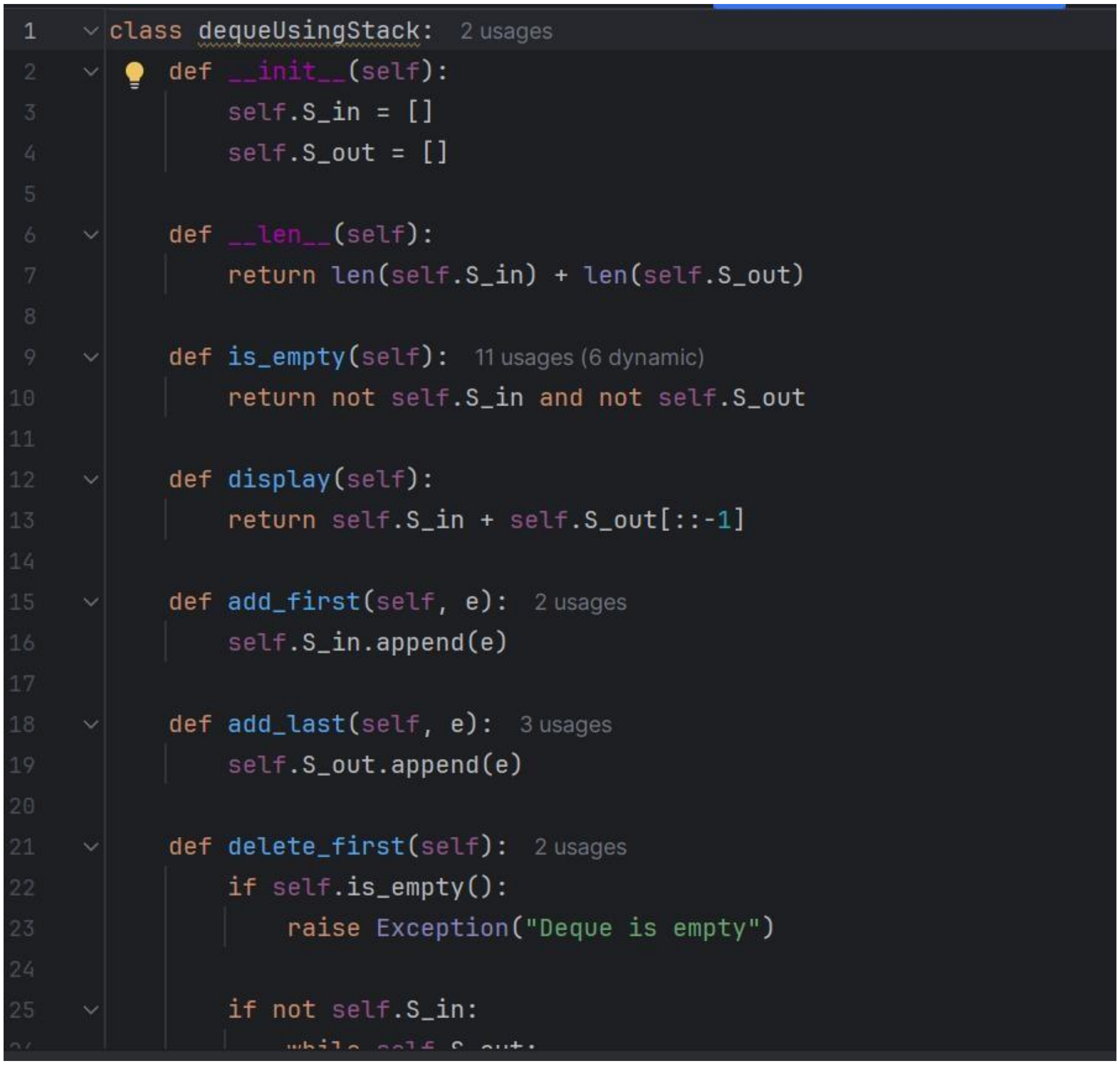


```
        if self.stack:
            return self.stack[-1]

        return self.queue.first()

    def last(self):  1 usage
        if self.is_empty():
            raise Exception("Deque is empty")

        if self.queue.is_empty():
            return self.stack[0]

        return self.queue.last()
```

UNIVERSITY OF Baguio

SCHOOL OF INFORMATION TECHNOLOGY

NAME: MARK LL OYD YADAO /RICHMOND BROQUEZA		
DSALGO1/IDB2	DATE SUBMITTED:12/062024	

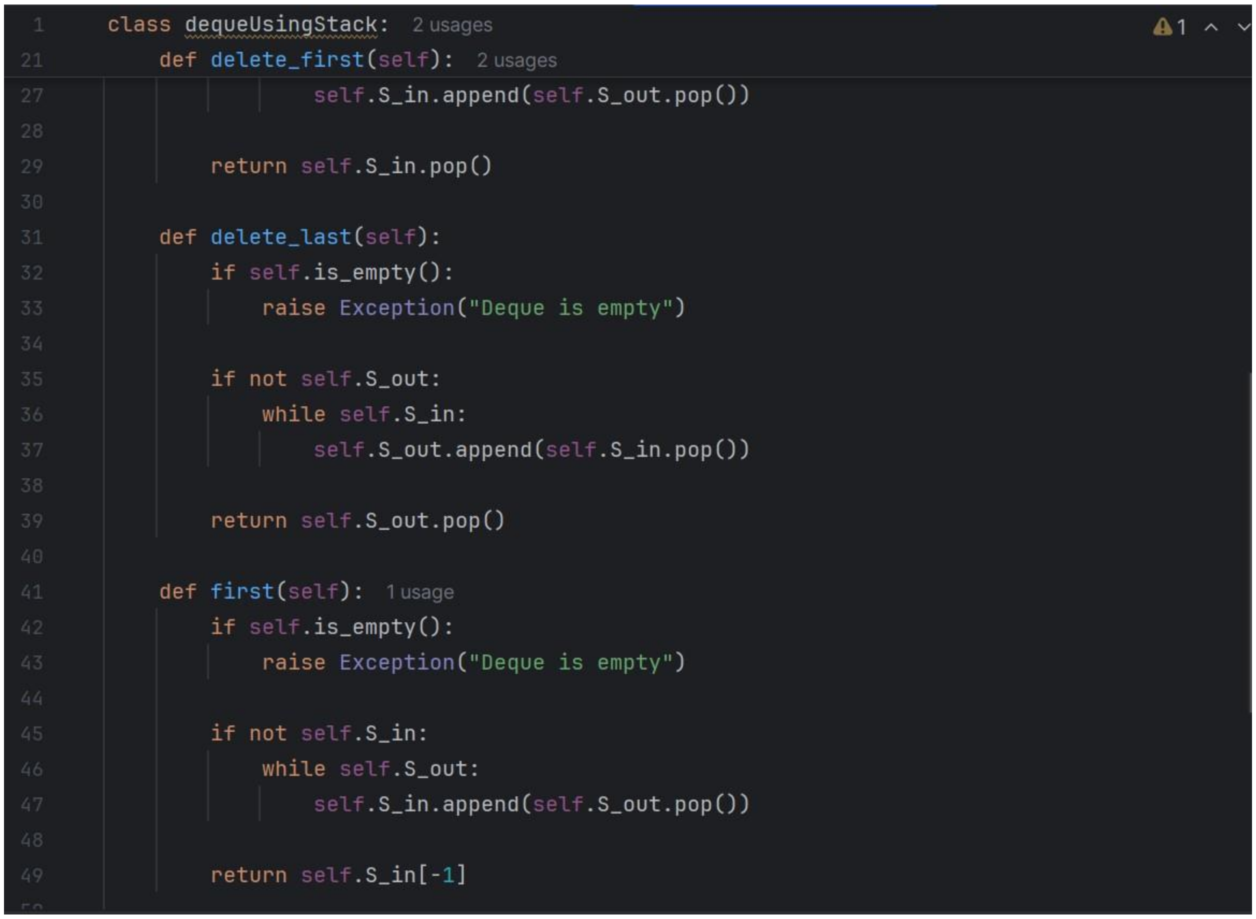


```
class dequeUsingStack:  2 usages
    def __init__(self):
        self.S_in = []
        self.S_out = []

    def __len__(self):
        return len(self.S_in) + len(self.S_out)

    def is_empty(self):  11 usages (6 dynamic)
        return not self.S_in and not self.S_out

    def display(self):
        return self.S_in + self.S_out[::-1]

    def add_first(self, e):  2 usages
        self.S_in.append(e)

    def add_last(self, e):  3 usages
        self.S_out.append(e)

    def delete_first(self):  2 usages
        if self.is_empty():
            raise Exception("Deque is empty")

        if not self.S_in:
```

UNIVERSITY OF Baguio

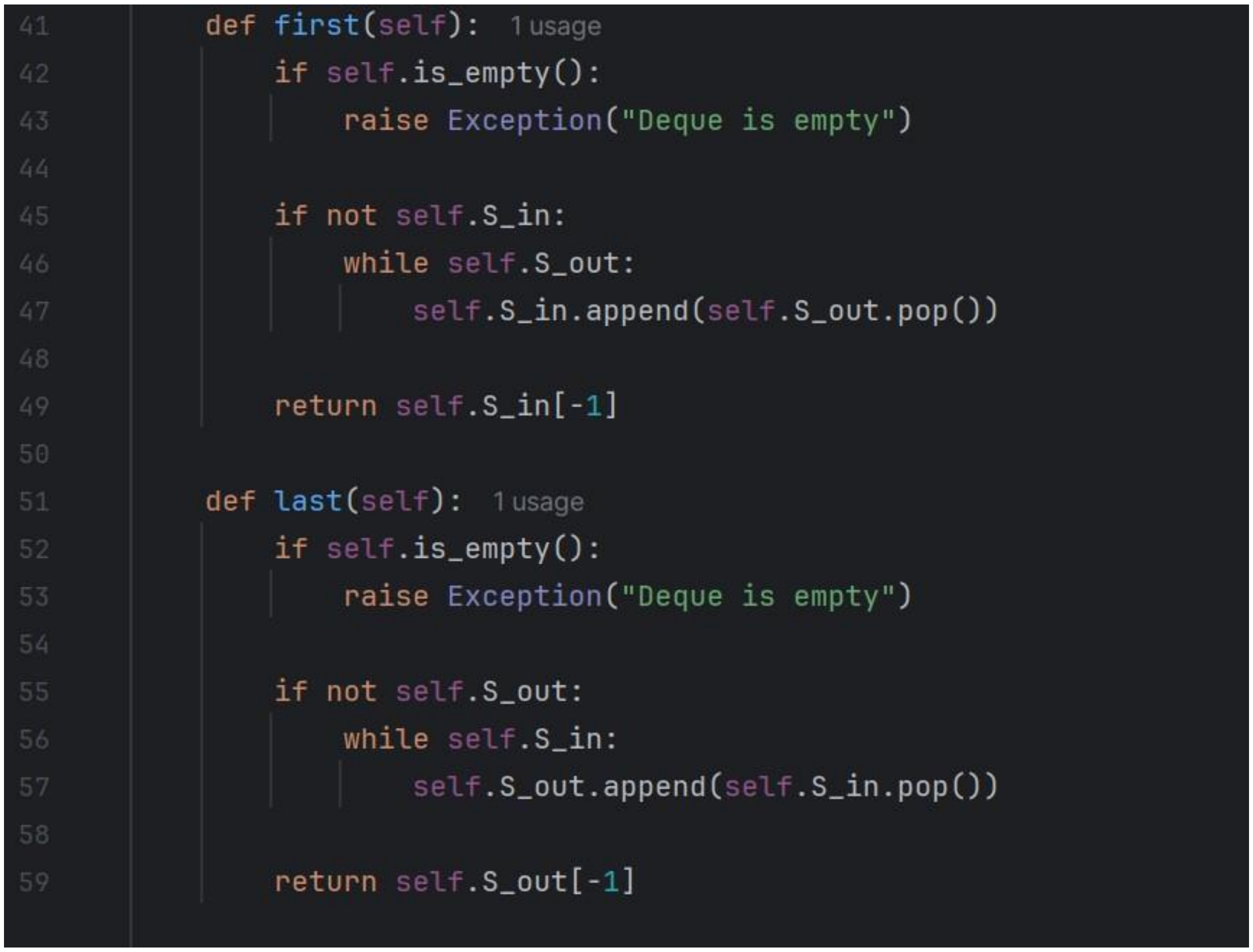
SCHOOL OF INFORMATION TECHNOLOGY

NAME: MARK LL OYD YADAO /RICHMOND BROQUEZA	
DSALGO1/IDB2	DATE SUBMITTED:12/062024



```
class dequeUsingStack:  2 usages
    def delete_first(self):  2 usages
                self.S_in.append(self.S_out.pop())

        return self.S_in.pop()

    def delete_last(self):
        if self.is_empty():
            raise Exception("Deque is empty")

        if not self.S_out:
            while self.S_in:
                self.S_out.append(self.S_in.pop())

        return self.S_out.pop()

    def first(self):  1 usage
        if self.is_empty():
            raise Exception("Deque is empty")

        if not self.S_in:
            while self.S_out:
                self.S_in.append(self.S_out.pop())

        return self.S_in[-1]
```

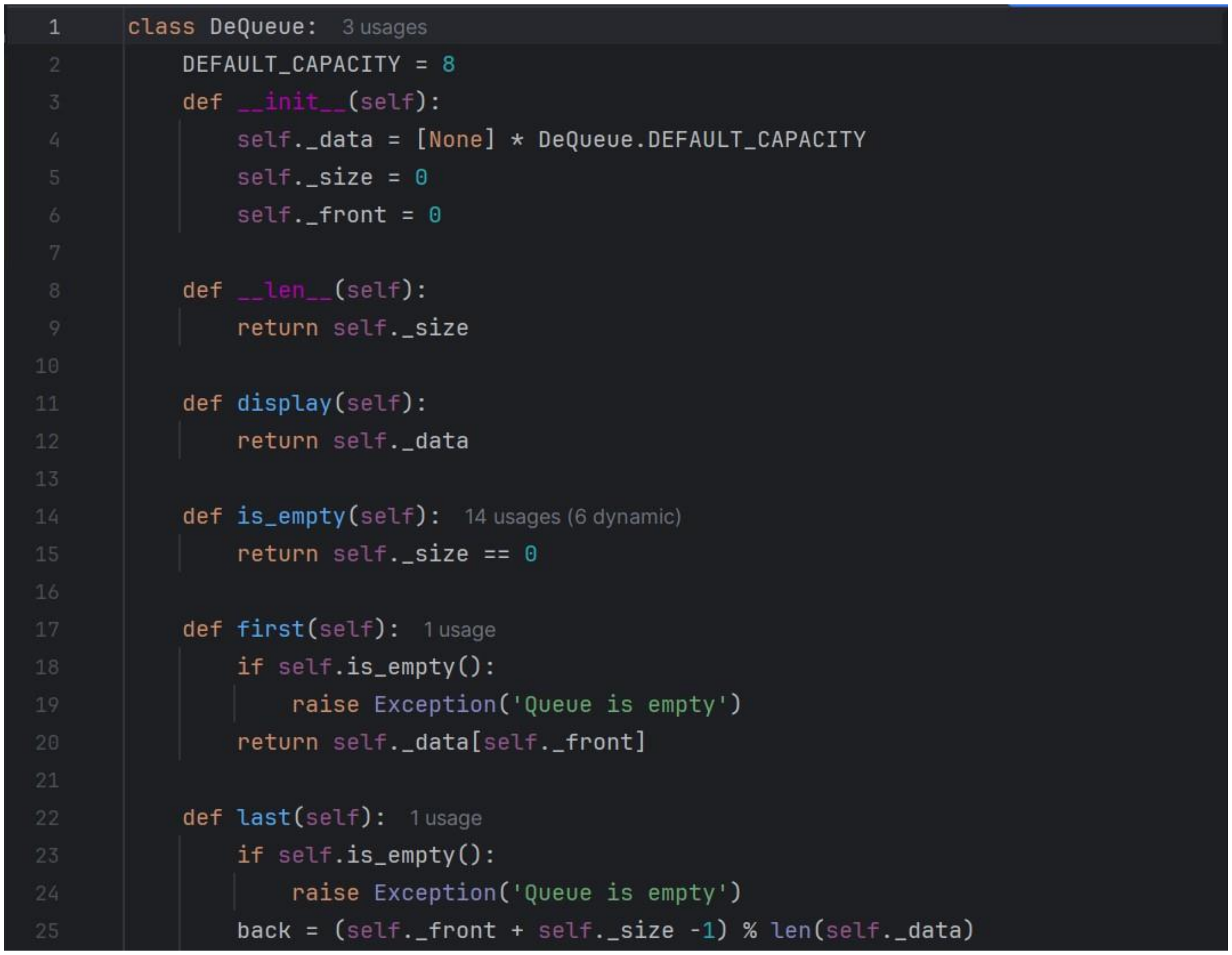
NAME: MARK LL OYD YADAO /RICHMOND BROQUEZA		
DSALGO1/IDB2	DATE SUBMITTED:12/062024	



```
    def first(self):  1 usage
        if self.is_empty():
            raise Exception("Deque is empty")

        if not self.S_in:
            while self.S_out:
                self.S_in.append(self.S_out.pop())

        return self.S_in[-1]

    def last(self):  1 usage
        if self.is_empty():
            raise Exception("Deque is empty")

        if not self.S_out:
            while self.S_in:
                self.S_out.append(self.S_in.pop())

        return self.S_out[-1]
```

SCHOOL OF INFORMATION TECHNOLOGY

NAME: MARK LL OYD YADAO /RICHMOND BROQUEZA	
DSALGO1/IDB2	DATE SUBMITTED:12/062024



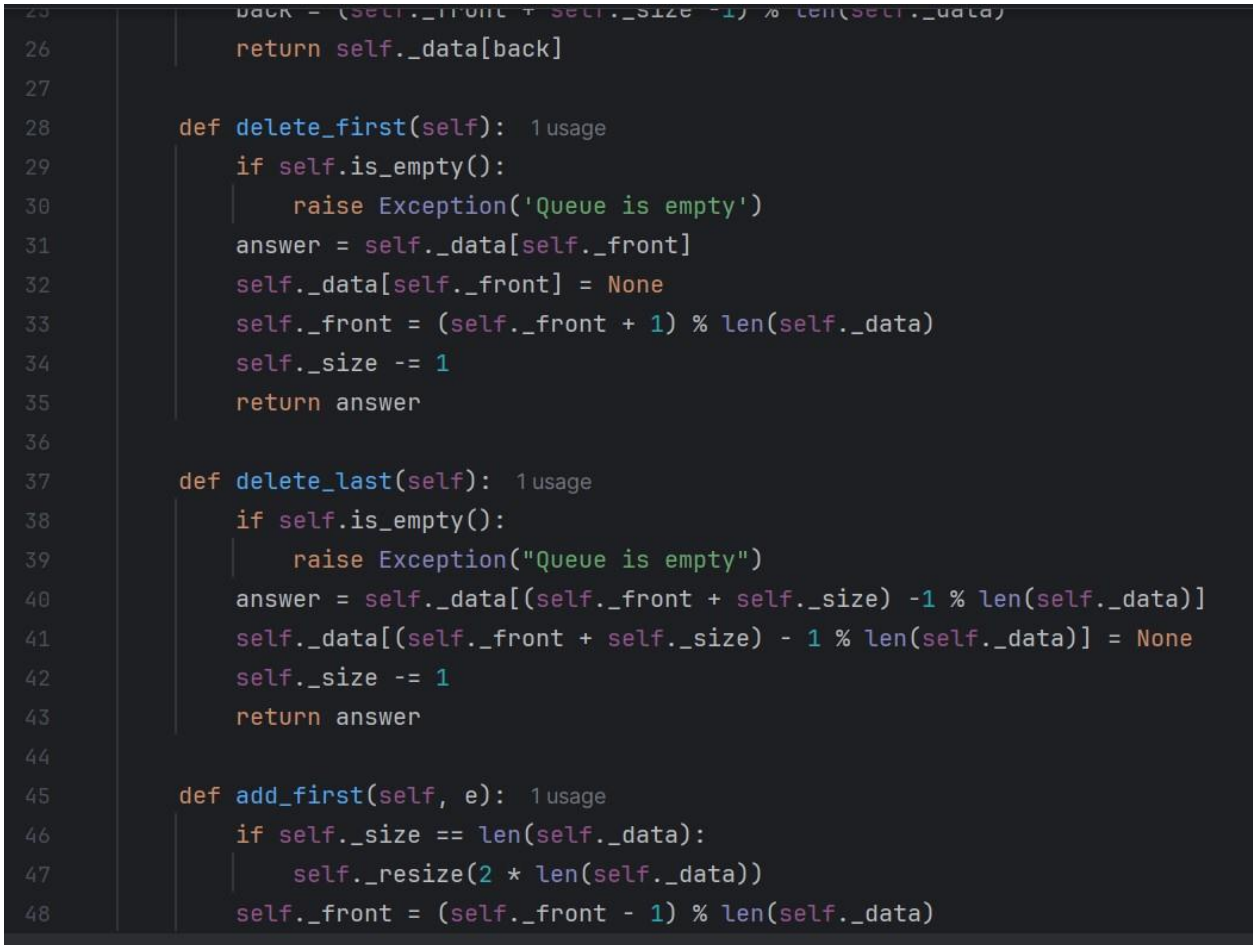
```
class DeQueue:  3 usages
    DEFAULT_CAPACITY = 8
    def __init__(self):
        self._data = [None] * DeQueue.DEFAULT_CAPACITY
        self._size = 0
        self._front = 0

    def __len__(self):
        return self._size

    def display(self):
        return self._data

    def is_empty(self):  14 usages (6 dynamic)
        return self._size == 0

    def first(self):  1 usage
        if self.is_empty():
            raise Exception('Queue is empty')
        return self._data[self._front]

    def last(self):  1 usage
        if self.is_empty():
            raise Exception('Queue is empty')
        back = (self._front + self._size -1) % len(self._data)
```

SECTION:IDB2 DATE SUBMITTED:12/062024

UNIVERSITY OF Baguio

SCHOOL OF INFORMATION TECHNOLOGY

NAME: MARK LL OYD YADAO /RICHMOND BROQUEZA		
DSALGO1/IDB2	DATE SUBMITTED:12/062024	

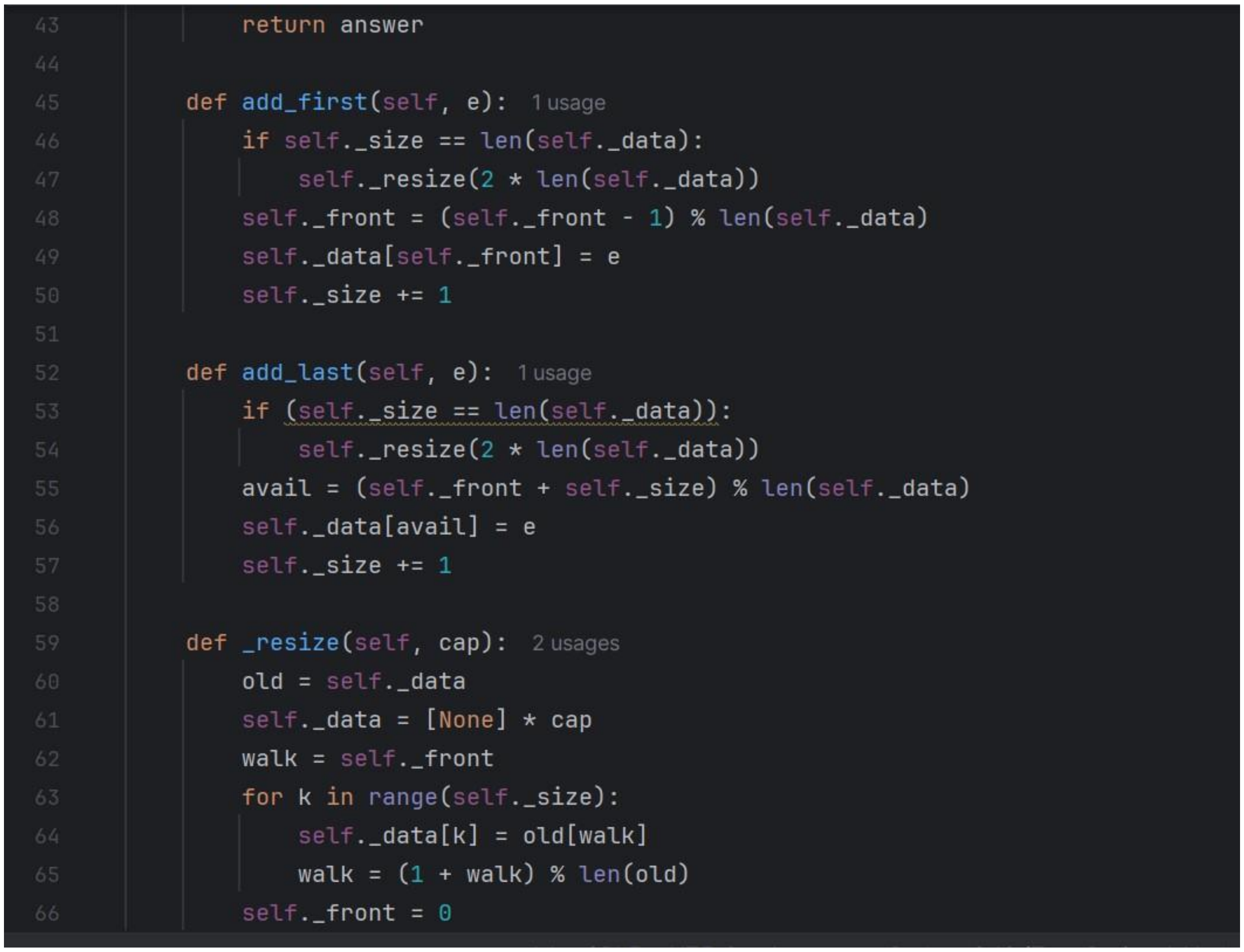


```
        back = (self._front + self._size -1) % len(self._data)
        return self._data[back]

    def delete_first(self):  1 usage
        if self.is_empty():
            raise Exception('Queue is empty')
        answer = self._data[self._front]
        self._data[self._front] = None
        self._front = (self._front + 1) % len(self._data)
        self._size -= 1
        return answer

    def delete_last(self):  1 usage
        if self.is_empty():
            raise Exception("Queue is empty")
        answer = self._data[(self._front + self._size) -1 % len(self._data)]
        self._data[(self._front + self._size) - 1 % len(self._data)] = None
        self._size -= 1
        return answer

    def add_first(self, e):  1 usage
        if self._size == len(self._data):
            self._resize(2 * len(self._data))
        self._front = (self._front - 1) % len(self._data)
```

UNIVERSITY OF Baguio

SCHOOL OF INFORMATION TECHNOLOGY

NAME: MARK LL OYD YADAO /RICHMOND BROQUEZA		
DSALGO1/IDB2	DATE SUBMITTED:12/062024	

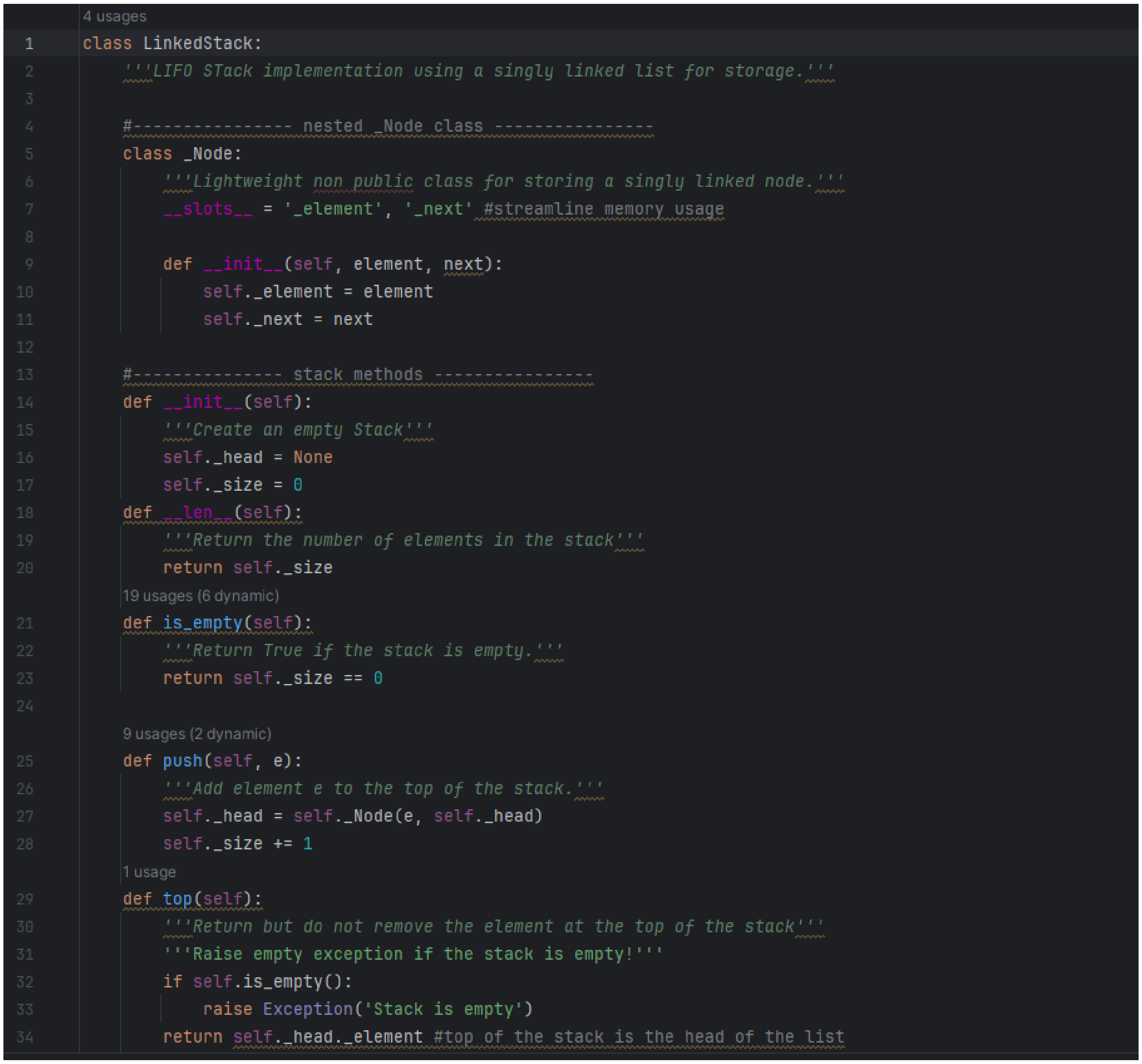


```
            return answer

    def add_first(self, e):  1 usage
        if self._size == len(self._data):
            self._resize(2 * len(self._data))
        self._front = (self._front - 1) % len(self._data)
        self._data[self._front] = e
        self._size += 1

    def add_last(self, e):  1 usage
        if (self._size == len(self._data)):
            self._resize(2 * len(self._data))
        avail = (self._front + self._size) % len(self._data)
        self._data[avail] = e
        self._size += 1

    def _resize(self, cap):  2 usages
        old = self._data
        self._data = [None] * cap
        walk = self._front
        for k in range(self._size):
            self._data[k] = old[walk]
            walk = (1 + walk) % len(old)
        self._front = 0
```

UNIVERSITY OF Baguio

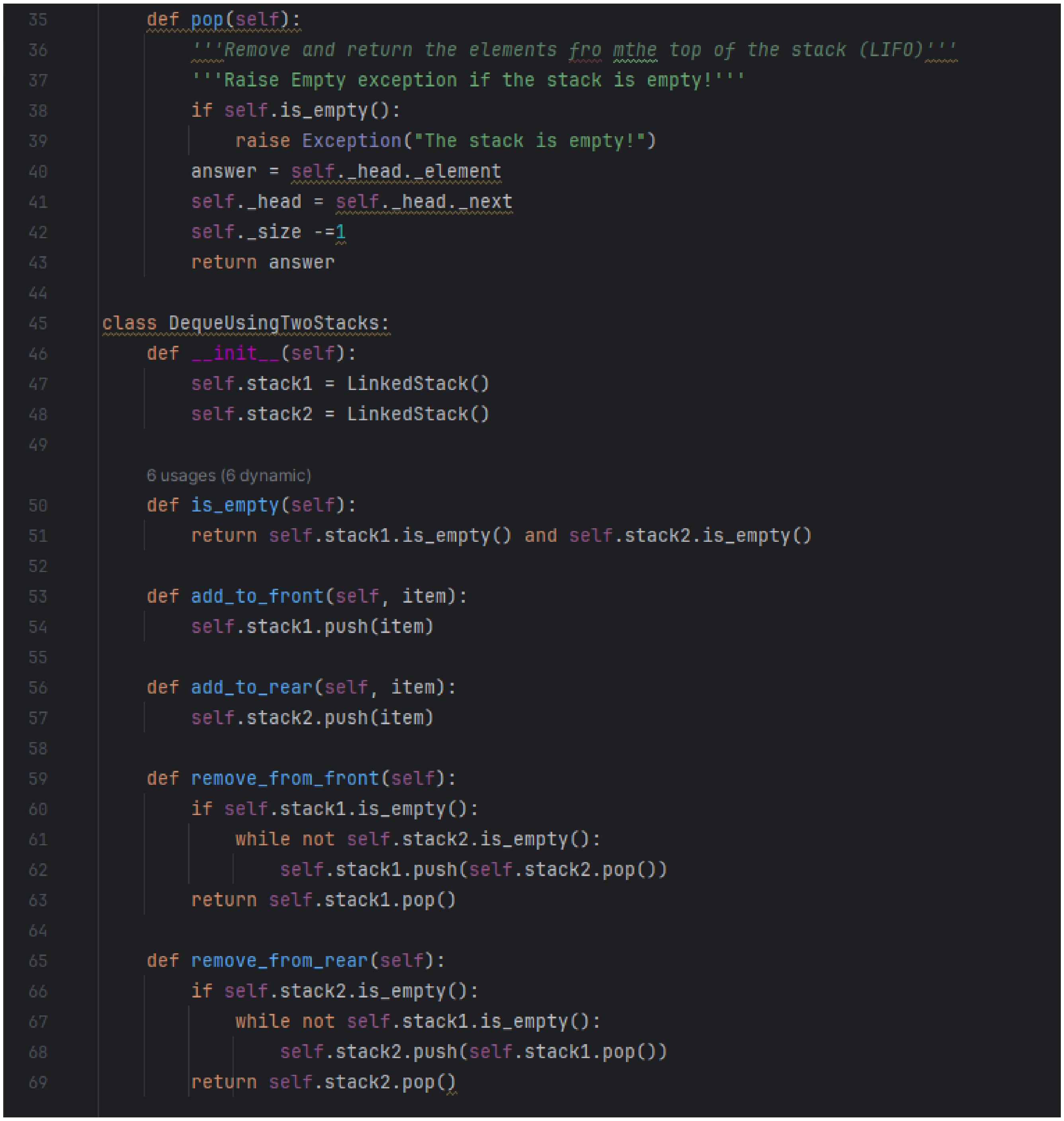
SCHOOL OF INFORMATION TECHNOLOGY

NAME: MARK LL OYD YADAO /RICHMOND BROQUEZA		
DSALGO1/IDB2	DATE SUBMITTED:12/062024	



```
4 usages
class LinkedStack:
    '''LIFO STack implementation using a singly linked list for storage.'''

    #---------------- nested _Node class ----------------
    class _Node:
        '''Lightweight non public class for storing a singly linked node.'''
        __slots__ = '_element', '_next' #streamline memory usage

        def __init__(self, element, next):
            self._element = element
            self._next = next

    #--------------- stack methods ----------------
    def __init__(self):
        '''Create an empty Stack'''
        self._head = None
        self._size = 0
    def __len__(self):
        '''Return the number of elements in the stack'''
        return self._size
    19 usages (6 dynamic)
    def is_empty(self):
        '''Return True if the stack is empty.'''
        return self._size == 0

    9 usages (2 dynamic)
    def push(self, e):
        '''Add element e to the top of the stack.'''
        self._head = self._Node(e, self._head)
        self._size += 1
    1 usage
    def top(self):
        '''Return but do not remove the element at the top of the stack'''
        '''Raise empty exception if the stack is empty!'''
        if self.is_empty():
            raise Exception('Stack is empty')
        return self._head._element #top of the stack is the head of the list
```

NAME: MARK LL OYD YADAO /RICHMOND BROQUEZA	
DSALGO1/IDB2	DATE SUBMITTED:12/062024

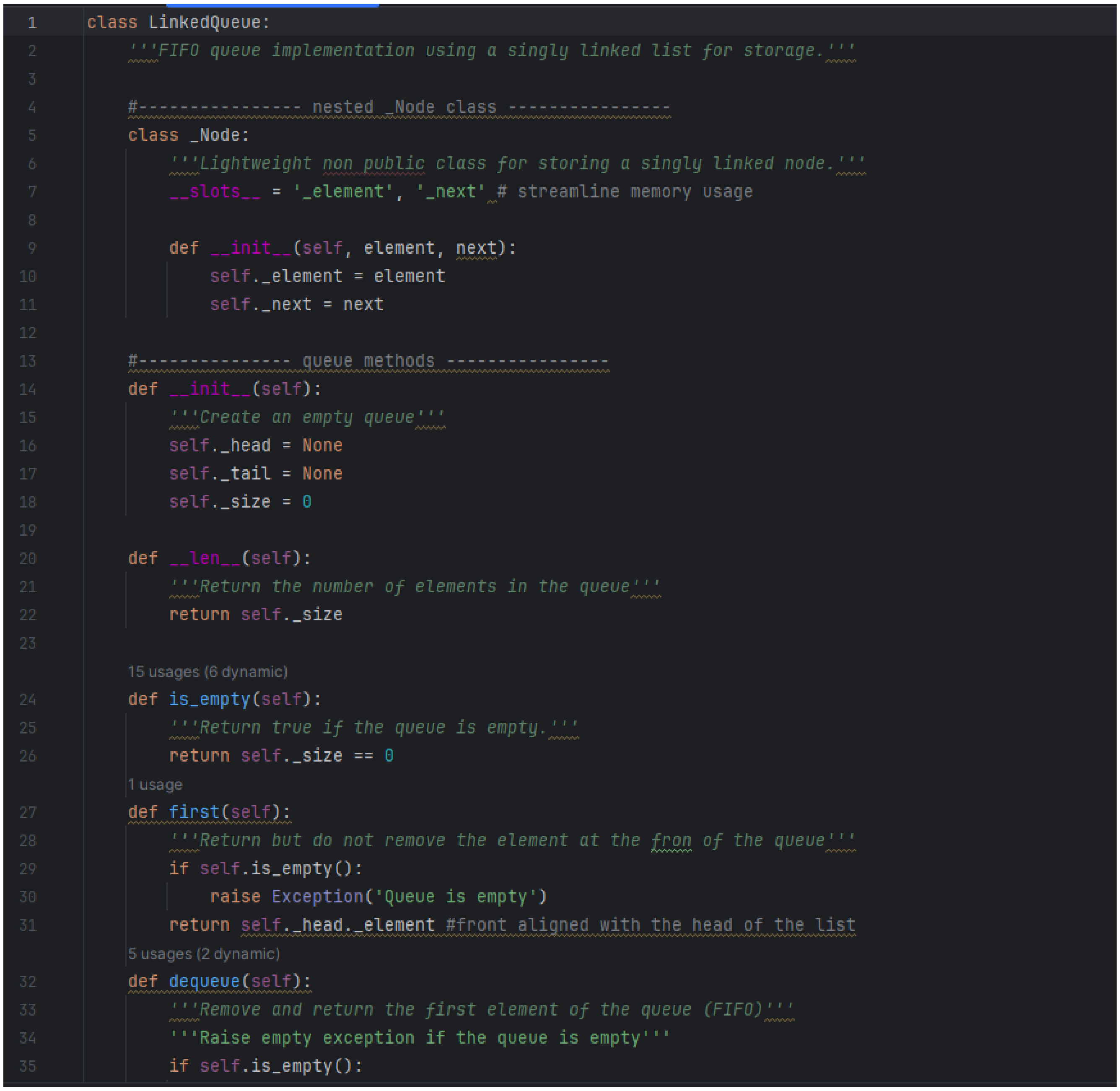


```
    def pop(self):
        '''Remove and return the elements fro mthe top of the stack (LIFO)'''
        '''Raise Empty exception if the stack is empty!'''
        if self.is_empty():
            raise Exception("The stack is empty!")
        answer = self._head._element
        self._head = self._head._next
        self._size -=1
        return answer

class DequeUsingTwoStacks:
    def __init__(self):
        self.stack1 = LinkedStack()
        self.stack2 = LinkedStack()

    6 usages (6 dynamic)
    def is_empty(self):
        return self.stack1.is_empty() and self.stack2.is_empty()

    def add_to_front(self, item):
        self.stack1.push(item)

    def add_to_rear(self, item):
        self.stack2.push(item)

    def remove_from_front(self):
        if self.stack1.is_empty():
            while not self.stack2.is_empty():
                self.stack1.push(self.stack2.pop())
        return self.stack1.pop()

    def remove_from_rear(self):
        if self.stack2.is_empty():
            while not self.stack1.is_empty():
                self.stack2.push(self.stack1.pop())
        return self.stack2.pop()
```

UNIVERSITY OF Baguio

SCHOOL OF INFORMATION TECHNOLOGY

NAME: MARK LL OYD YADAO /RICHMOND BROQUEZA		
DSALGO1/IDB2	DATE SUBMITTED:12/062024	

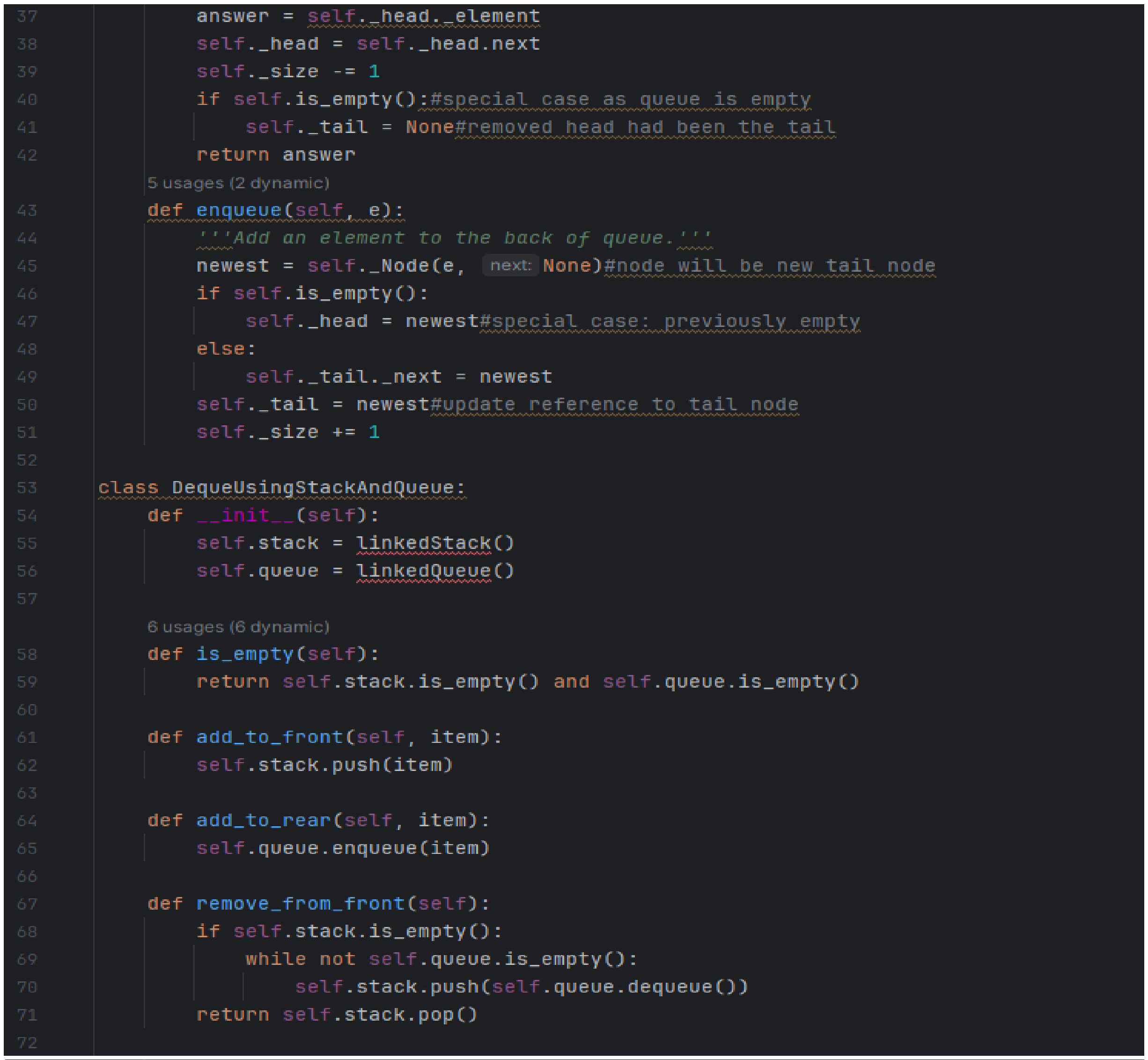


```
class LinkedQueue:
    '''FIFO queue implementation using a singly linked list for storage.'''

    #---------------- nested _Node class ----------------
    class _Node:
        '''Lightweight non public class for storing a singly linked node.'''
        __slots__ = '_element', '_next' # streamline memory usage

        def __init__(self, element, next):
            self._element = element
            self._next = next

    #--------------- queue methods ----------------
    def __init__(self):
        '''Create an empty queue'''
        self._head = None
        self._tail = None
        self._size = 0

    def __len__(self):
        '''Return the number of elements in the queue'''
        return self._size

    15 usages (6 dynamic)
    def is_empty(self):
        '''Return true if the queue is empty.'''
        return self._size == 0
    1 usage
    def first(self):
        '''Return but do not remove the element at the fron of the queue'''
        if self.is_empty():
            raise Exception('Queue is empty')
        return self._head._element #front aligned with the head of the list
    5 usages (2 dynamic)
    def dequeue(self):
        '''Remove and return the first element of the queue (FIFO)'''
        '''Raise empty exception if the queue is empty'''
        if self.is_empty():
```

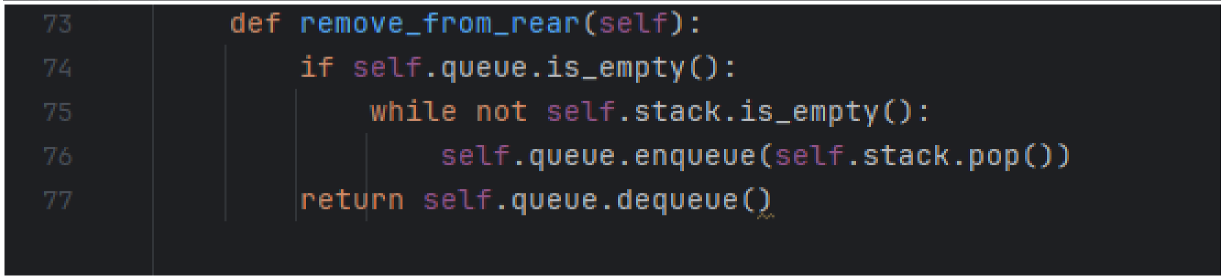
UNIVERSITY OF Baguio

SCHOOL OF INFORMATION TECHNOLOGY

NAME: MARK LL OYD YADAO /RICHMOND BROQUEZA	
DSALGO1/IDB2	DATE SUBMITTED:12/062024



```
        answer = self._head._element
        self._head = self._head.next
        self._size -= 1
        if self.is_empty():#special case as queue is empty
            self._tail = None#removed head had been the tail
        return answer
    5 usages (2 dynamic)
    def enqueue(self, e):
        '''Add an element to the back of queue.'''
        newest = self._Node(e, next: None)#node will be new tail node
        if self.is_empty():
            self._head = newest#special case: previously empty
        else:
            self._tail._next = newest
        self._tail = newest#update reference to tail node
        self._size += 1

class DequeUsingStackAndQueue:
    def __init__(self):
        self.stack = linkedStack()
        self.queue = linkedQueue()

    6 usages (6 dynamic)
    def is_empty(self):
        return self.stack.is_empty() and self.queue.is_empty()

    def add_to_front(self, item):
        self.stack.push(item)

    def add_to_rear(self, item):
        self.queue.enqueue(item)

    def remove_from_front(self):
        if self.stack.is_empty():
            while not self.queue.is_empty():
                self.stack.push(self.queue.dequeue())
        return self.stack.pop()

```

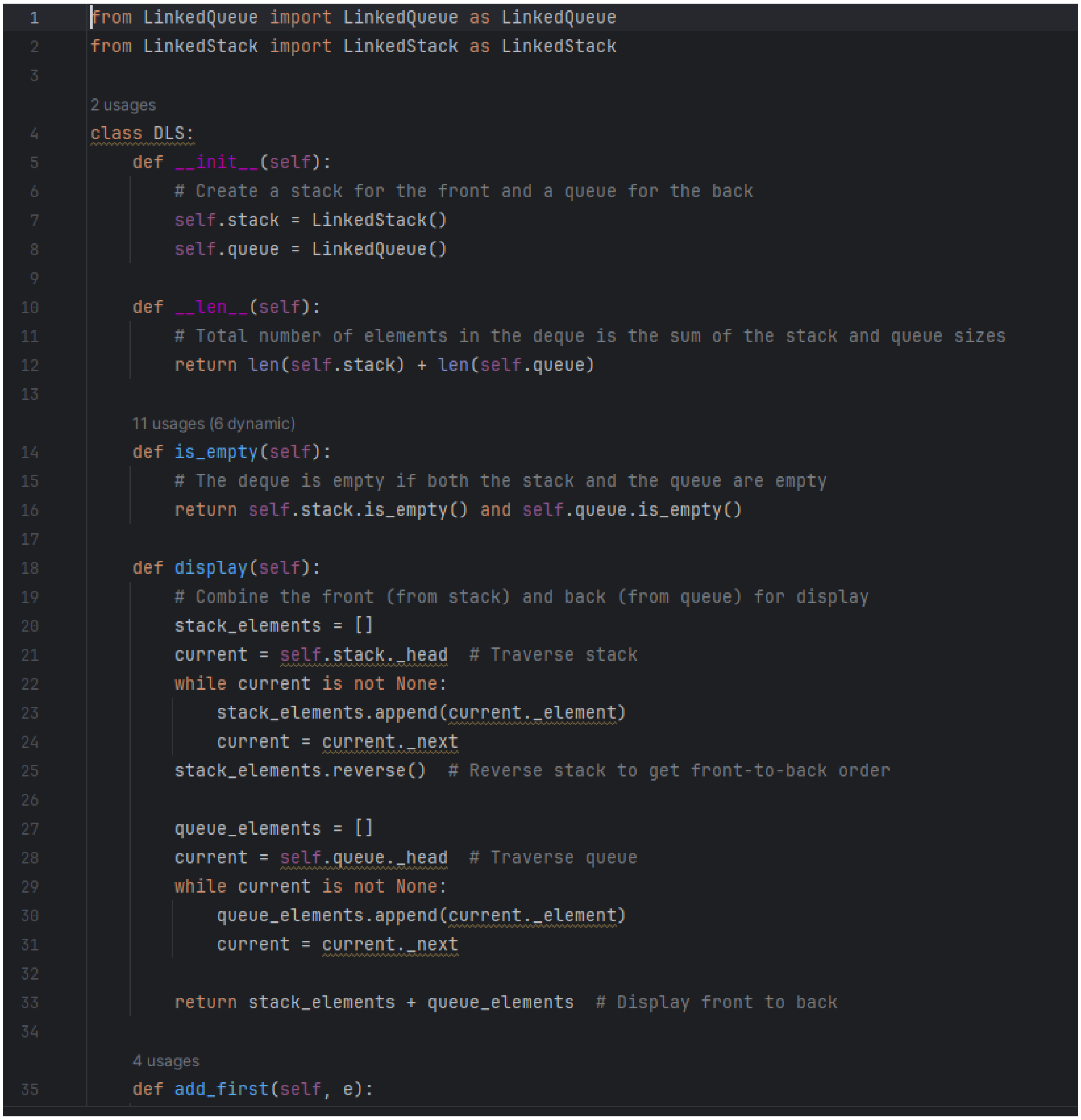


```
    def remove_from_rear(self):
        if self.queue.is_empty():
            while not self.stack.is_empty():
                self.queue.enqueue(self.stack.pop())
        return self.queue.dequeue()
```

UNIVERSITY OF Baguio

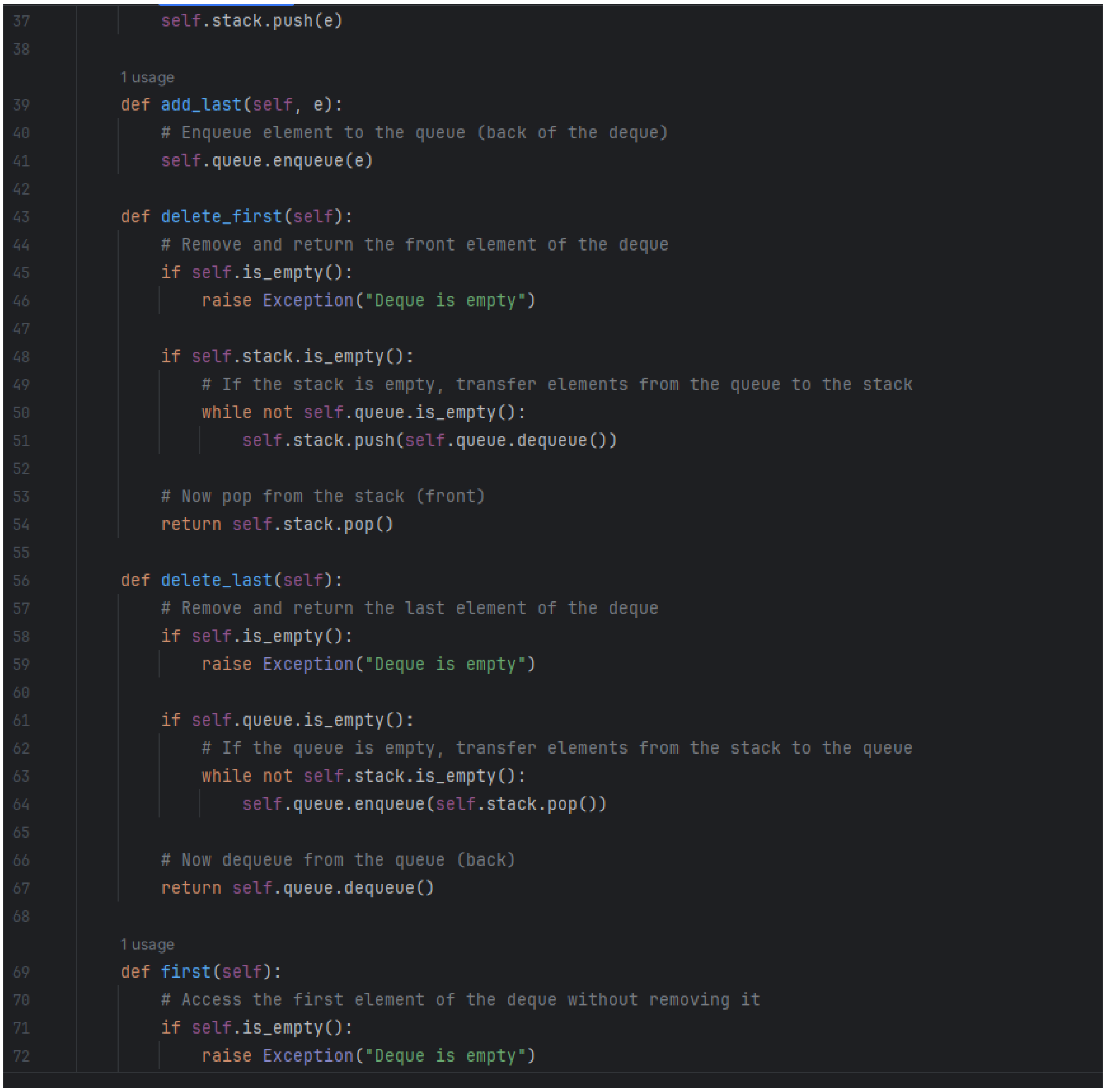
SCHOOL OF INFORMATION TECHNOLOGY

NAME: MARK LL OYD YADAO /RICHMOND BROQUEZA		
DSALGO1/IDB2	DATE SUBMITTED:12/062024	



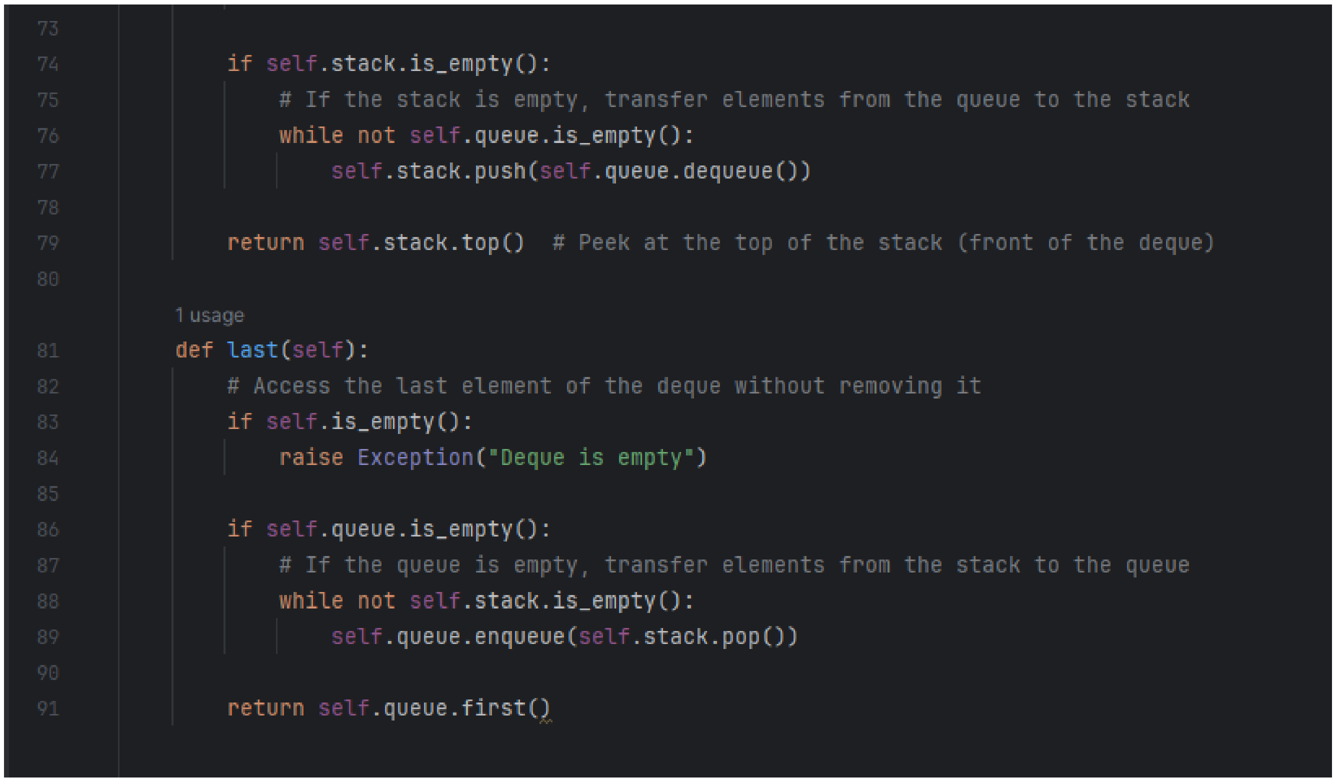
```
from LinkedQueue import LinkedQueue as LinkedQueue
from LinkedStack import LinkedStack as LinkedStack

2 usages
class DLS:
    def __init__(self):
        # Create a stack for the front and a queue for the back
        self.stack = LinkedStack()
        self.queue = LinkedQueue()

    def __len__(self):
        # Total number of elements in the deque is the sum of the stack and queue sizes
        return len(self.stack) + len(self.queue)

    11 usages (6 dynamic)
    def is_empty(self):
        # The deque is empty if both the stack and the queue are empty
        return self.stack.is_empty() and self.queue.is_empty()

    def display(self):
        # Combine the front (from stack) and back (from queue) for display
        stack_elements = []
        current = self.stack._head  # Traverse stack
        while current is not None:
            stack_elements.append(current._element)
            current = current._next
        stack_elements.reverse()  # Reverse stack to get front-to-back order

        queue_elements = []
        current = self.queue._head  # Traverse queue
        while current is not None:
            queue_elements.append(current._element)
            current = current._next

        return stack_elements + queue_elements  # Display front to back

    4 usages
    def add_first(self, e):
```

NAME: MARK LL OYD YADAO /RICHMOND BROQUEZA		
DSALGO1/IDB2	DATE SUBMITTED:12/062024	



```
        self.stack.push(e)

    1 usage
    def add_last(self, e):
        # Enqueue element to the queue (back of the deque)
        self.queue.enqueue(e)

    def delete_first(self):
        # Remove and return the front element of the deque
        if self.is_empty():
            raise Exception("Deque is empty")

        if self.stack.is_empty():
            # If the stack is empty, transfer elements from the queue to the stack
            while not self.queue.is_empty():
                self.stack.push(self.queue.dequeue())

        # Now pop from the stack (front)
        return self.stack.pop()

    def delete_last(self):
        # Remove and return the last element of the deque
        if self.is_empty():
            raise Exception("Deque is empty")

        if self.queue.is_empty():
            # If the queue is empty, transfer elements from the stack to the queue
            while not self.stack.is_empty():
                self.queue.enqueue(self.stack.pop())

        # Now dequeue from the queue (back)
        return self.queue.dequeue()

    1 usage
    def first(self):
        # Access the first element of the deque without removing it
        if self.is_empty():
            raise Exception("Deque is empty")
```

NAME: MARK LL OYD YADAO /RICHMOND BROQUEZA	
DSALGO1/IDB2	DATE SUBMITTED:12/062024

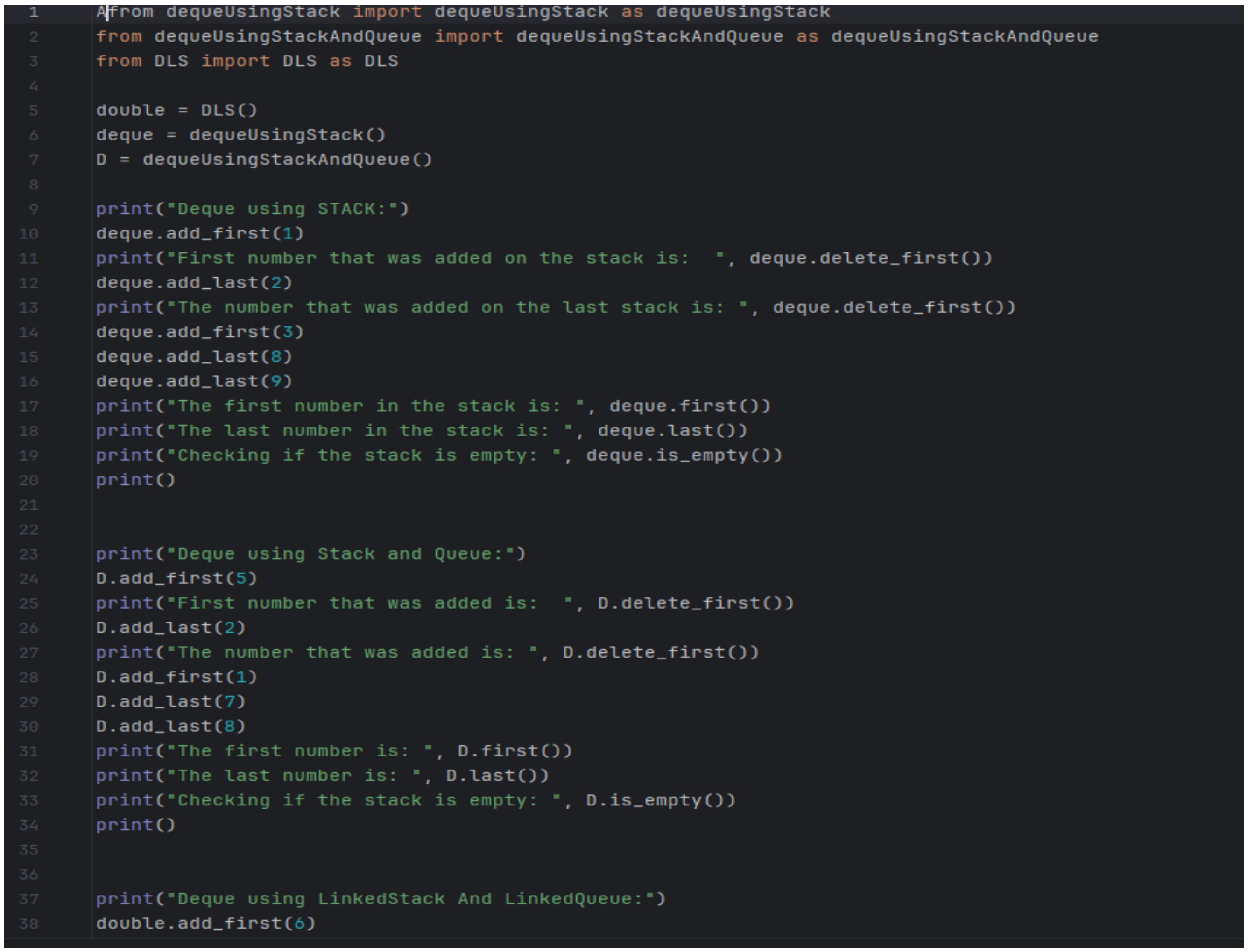


```python
        if self.stack.is_empty():
            # If the stack is empty, transfer elements from the queue to the stack
            while not self.queue.is_empty():
                self.stack.push(self.queue.dequeue())

        return self.stack.top()  # Peek at the top of the stack (front of the deque)

    1 usage
    def last(self):
        # Access the last element of the deque without removing it
        if self.is_empty():
            raise Exception("Deque is empty")

        if self.queue.is_empty():
            # If the queue is empty, transfer elements from the stack to the queue
            while not self.stack.is_empty():
                self.queue.enqueue(self.stack.pop())

        return self.queue.first()
```

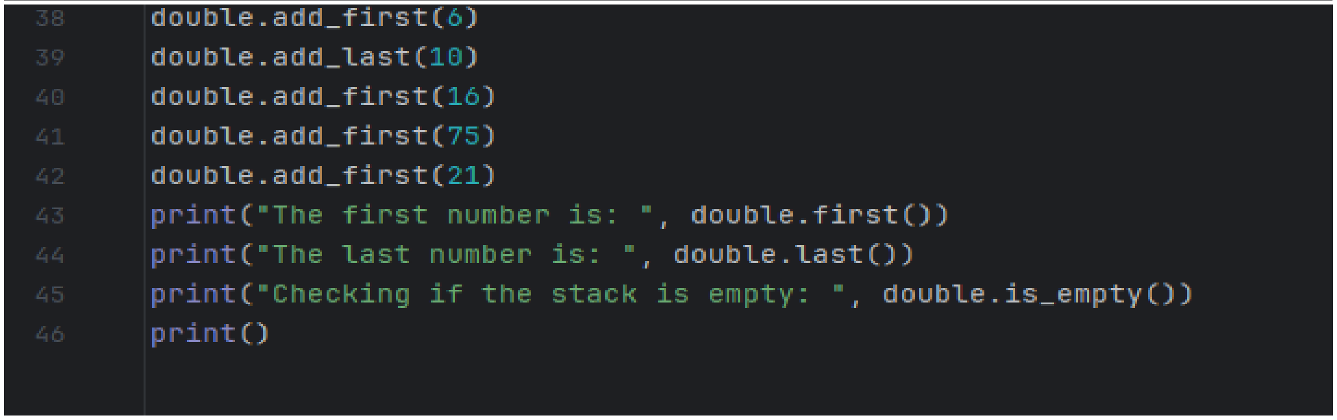
UNIVERSITY OF Baguio

SCHOOL OF INFORMATION TECHNOLOGY

NAME: MARK LL OYD YADAO /RICHMOND BROQUEZA		
DSALGO1/IDB2	DATE SUBMITTED:12/062024	



```python
from dequeUsingStack import dequeUsingStack as dequeUsingStack
from dequeUsingStackAndQueue import dequeUsingStackAndQueue as dequeUsingStackAndQueue
from DLS import DLS as DLS

double = DLS()
deque = dequeUsingStack()
D = dequeUsingStackAndQueue()

print("Deque using STACK:")
deque.add_first(1)
print("First number that was added on the stack is:  ", deque.delete_first())
deque.add_last(2)
print("The number that was added on the last stack is: ", deque.delete_first())
deque.add_first(3)
deque.add_last(8)
deque.add_last(9)
print("The first number in the stack is: ", deque.first())
print("The last number in the stack is: ", deque.last())
print("Checking if the stack is empty: ", deque.is_empty())
print()


print("Deque using Stack and Queue:")
D.add_first(5)
print("First number that was added is:  ", D.delete_first())
D.add_last(2)
print("The number that was added is: ", D.delete_first())
D.add_first(1)
D.add_last(7)
D.add_last(8)
print("The first number is: ", D.first())
print("The last number is: ", D.last())
print("Checking if the stack is empty: ", D.is_empty())
print()


print("Deque using LinkedStack And LinkedQueue:")
double.add_first(6)
```



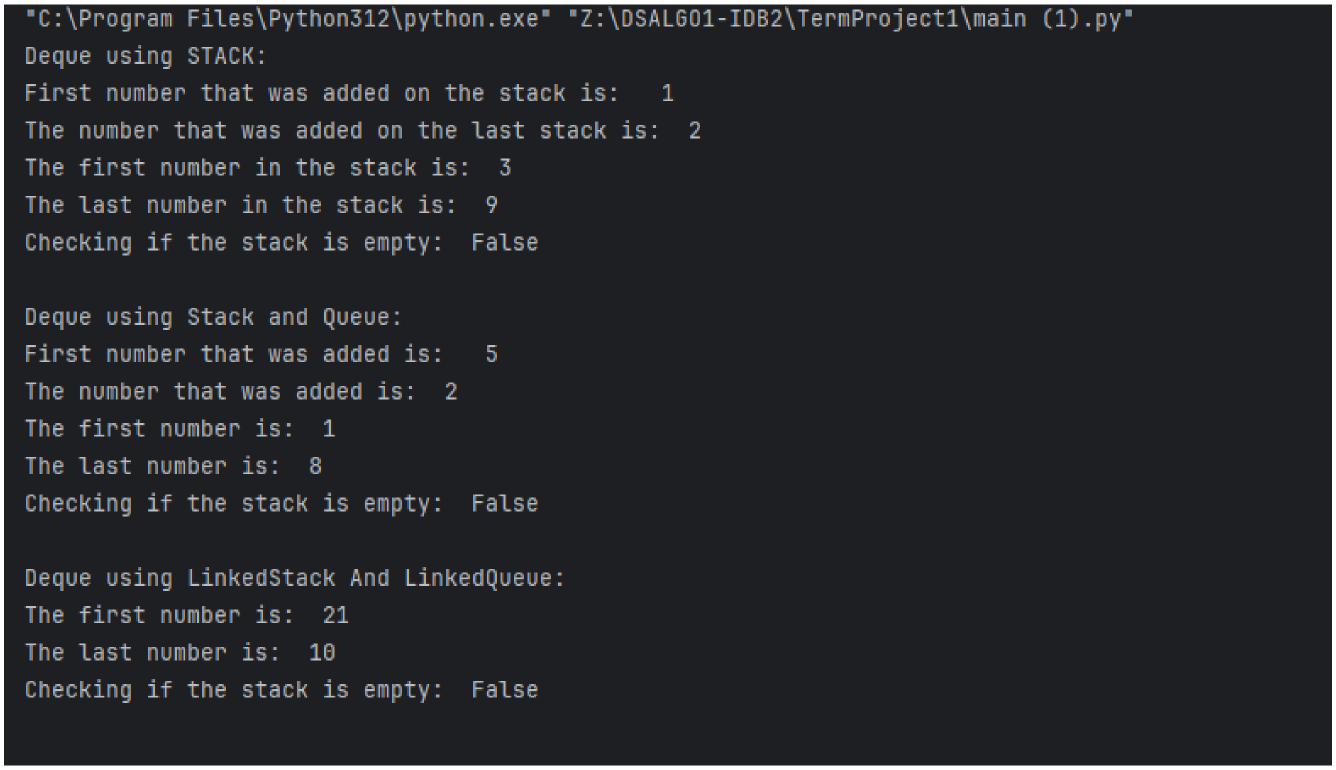
```python
double.add_first(6)
double.add_last(10)
double.add_first(16)
double.add_first(75)
double.add_first(21)
print("The first number is: ", double.first())
print("The last number is: ", double.last())
print("Checking if the stack is empty: ", double.is_empty())
print()
```

UNIVERSITY OF Baguio

SCHOOL OF INFORMATION TECHNOLOGY

NAME: MARK LL OYD YADAO /RICHMOND BROQUEZA		
DSALGO1/IDB2	DATE SUBMITTED:12/062024	

OUTPUT



```
"C:\Program Files\Python312\python.exe" "Z:\DSALGO1-IDB2\TermProject1\main (1).py"
Deque using STACK:
First number that was added on the stack is:   1
The number that was added on the last stack is:  2
The first number in the stack is:  3
The last number in the stack is:  9
Checking if the stack is empty:  False

Deque using Stack and Queue:
First number that was added is:   5
The number that was added is:  2
The first number is:  1
The last number is:  8
Checking if the stack is empty:  False

Deque using LinkedStack And LinkedQueue:
The first number is:  21
The last number is:  10
Checking if the stack is empty:  False
```