[4]

```python
class Student:
    def __init__(self, Masv, Hoten=None, Malop=None, DiemTB=None):
        self.Masv = Masv
        self.Hoten = Hoten
        self.Malop= Malop
        self.DiemTB= DiemTB

    def __str__(self):
        return f"MaSV: {self.Masv}, HoTen: {self.HoTen}"
```

[-]

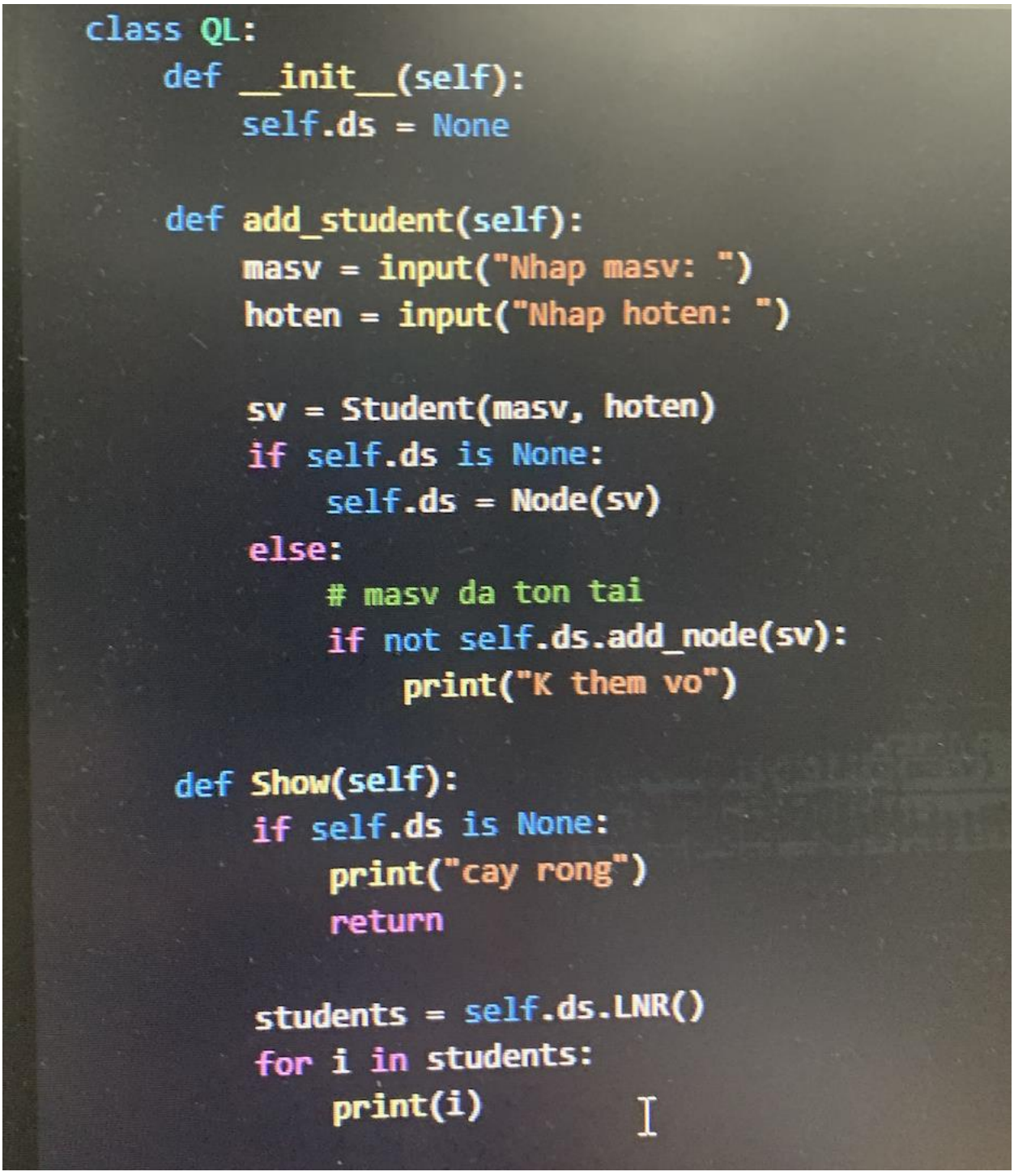
```python
class Node:
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None

    def add_node(self, student):
        if student == self.data:
            return False

        if student < self.data:
            if self.left is None:
                self.left = Node(student)
                return True
            else:
                self.left.add_node(student)

        elif student > self.data:
            if self.right is None:
                self.right = Node(student)
                return True
            else:
                self.right.add_node(student)
```

```
def LNR(self):
    if self.left:
        self.left.LNR()

    print(self.data)

    if self.right:
        self.right.LNR()

def countLeaf(self):
    dem = 0
    if self.left is None and self.right is No
        return 1

    if self.left:
        dem += self.left.countLeaf()

    if self.right:
        dem +=  self.right.countLeaf()

    return dem
```

```python
def heightTree(self):
    height_left =0
    height_right =0

    if self.left:
        height_left = self.left.heightTree()

    if self.right:
        height_right = self.right.heightTree()

    return max(height_left, height_right) + 1
```

```python
def search(self, masv):
    if self.data.masv==masv:
        return self

    if masv < self.data.masv and self.left:
        return self.left.search( masv)

    if masv > self.data.masv and self.right:
        return self.right.search( masv)
```

```
def delete(self, masv):
    if masv < self.data.masv and self.left:
        self.left = self.left.delete(masv)

    elif masv > self.data.masv and self.right:
        self.right = self.right.delete(masv)

    else:
        # case1: nut la
        if self.left is None and self.right is None:
            return None
        # case2: nut co 1 con
        if self.left is None:
            return self.right
        if self.right is None:
            return self.left

        # case 3: nut co 2 con
        temp = self.right # 5
        while self.left:
            temp = self.left

        self.data = temp.data

        self.right = self.right.delete(temp.data.masv)

    return self
```



```
class QL:
    def __init__(self):
        self.ds = None

    def add_student(self):
        masv = input("Nhap masv: ")
        hoten = input("Nhap hoten: ")

        sv = Student(masv, hoten)
        if self.ds is None:
            self.ds = Node(sv)
        else:
            # masv da ton tai
            if not self.ds.add_node(sv):
                print("K them vo")

    def Show(self):
        if self.ds is None:
            print("cay rong")
            return

        students = self.ds.LNR()
        for i in students:
            print(i)
```

BÀI CŨ TUẦN CŨ CŨ TRƯỚC

```
def find_even_numbers(self):
    result = []
    if self.data % 2 == 0:
        result.append(self.data)

    if self.left:
        result.extend(self.left.find_even_numbers())

    if self.right:
        result.extend(self.right.find_even_numbers())

    return result

def find_min(self):
    if self.left is None:
        return self.data
    return self.left.find_min()

def find_max(self):
    if self.right is None:
        return self.data

    return self.right.find_max()
```