

Nama : Rayhan Nurfalah Lukman

NIM : L200180100

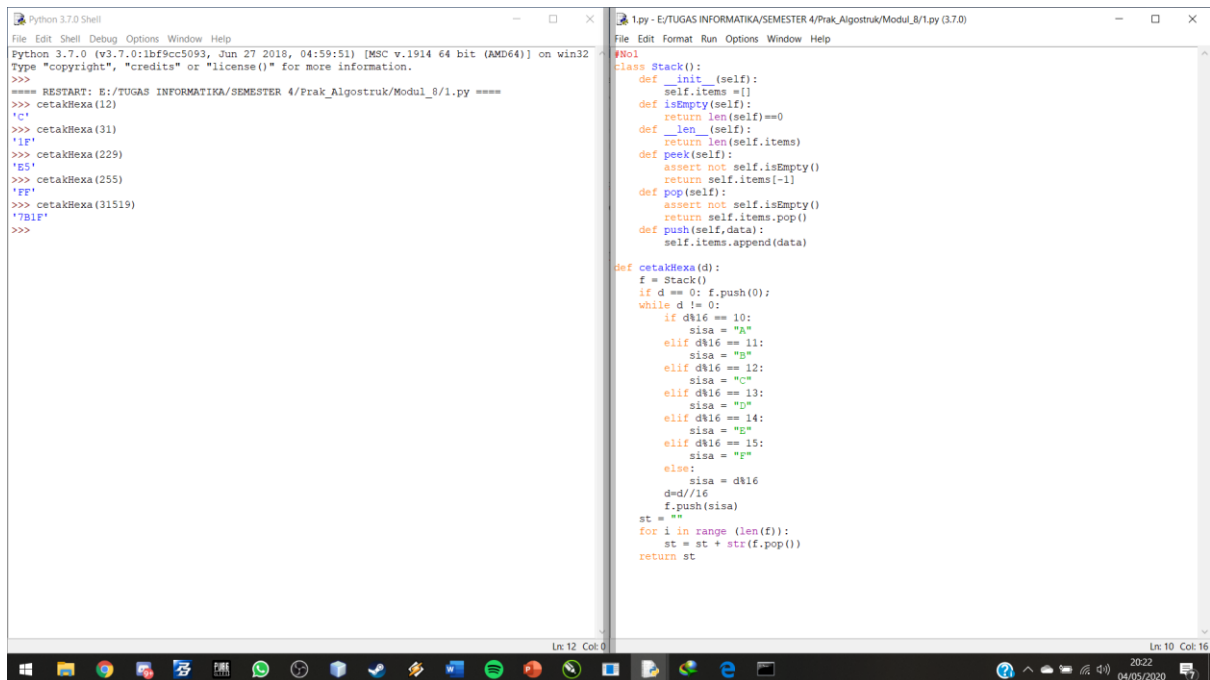
Kelas : D

Praktikum ASD

Modul 8

Tugas

1. Buatlah Program untuk mengubah representasi suatu bilangan dari basis sepuluh ke basis dua.



```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2019, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: E:/TUGAS INFORMATIKA/SEMESTER 4/Prak_Algostruk/Modul_8/1.py ====
>>> cetakHexa(12)
'c'
>>> cetakHexa(31)
'1f'
>>> cetakHexa(229)
'b5'
>>> cetakHexa(255)
'ff'
>>> cetakHexa(31519)
'7b1f'
>>>
```

```
1.py - E:/TUGAS INFORMATIKA/SEMESTER 4/Prak_Algostruk/Modul_8/1.py (3.7.0)
File Edit Format Run Options Window Help
#01
class Stack():
    def __init__(self):
        self.items = []
    def isEmpty(self):
        return len(self) == 0
    def __len__(self):
        return len(self.items)
    def peek(self):
        assert not self.isEmpty()
        return self.items[-1]
    def pop(self):
        assert not self.isEmpty()
        return self.items.pop()
    def push(self, data):
        self.items.append(data)

def cetakHexa(d):
    f = Stack()
    if d == 0: f.push(0)
    while d != 0:
        if d%16 == 10:
            sisa = "a"
        elif d%16 == 11:
            sisa = "b"
        elif d%16 == 12:
            sisa = "c"
        elif d%16 == 13:
            sisa = "d"
        elif d%16 == 14:
            sisa = "e"
        elif d%16 == 15:
            sisa = "f"
        else:
            sisa = d%16
        d = d//16
        f.push(sisa)
    st = ""
    for i in range(len(f)):
        st = st + str(f.pop())
    return st
```

2. Eksekusi program berikut dengan pensil dan kertas, dan tunjukkan isi stack-nya pada setiap Langkah.

The screenshot shows a Python 3.7.0 Shell window on the left and a Python script editor on the right. The shell window displays the execution of a program that pushes numbers 0 to 15 onto a stack and then pops them. The script editor shows the implementation of a Stack class with methods: `__init__`, `isEmpty`, `len`, `peek`, `pop`, and `push`. The stack is implemented as a list.

```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (tags/v3.7.0:1bf9cc5093, Jun 27 2019, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:/TUGAS INFORMATIKA/SEMESTER 4/Prak_Algostruk/Modul_8/2.py =====
>>> nilai = Stack()
>>> for i in range(16):
>>>     if i % 3 == 0:
>>>         nilai.push(i)
>>>
>>> nilai.pop()
15
>>> nilai.pop()
12
>>> nilai.pop()
9
>>> nilai.pop()
6
>>> nilai.pop()
3
>>> nilai.pop()
0
>>>
```

```
3.py - E:/TUGAS INFORMATIKA/SEMESTER 4/Prak_Algostruk/Modul_8/2.py (3.7.0)
File Edit Format Run Options Window Help
#No2
class Stack():
    def __init__(self):
        self.items = []
    def isEmpty(self):
        return len(self) == 0
    def __len__(self):
        return len(self.items)
    def peek(self):
        assert not self.isEmpty()
        return self.items[-1]
    def pop(self):
        assert not self.isEmpty()
        return self.items.pop()
    def push(self, data):
        self.items.append(data)

nilai = Stack()
for i in range(16):
    if i % 3 == 0:
        nilai.push(i)
```

3. Eksekusi program berikut dengan pensil dan kertas, dan tunjukkan isi stack-nya pada setiap Langkah.

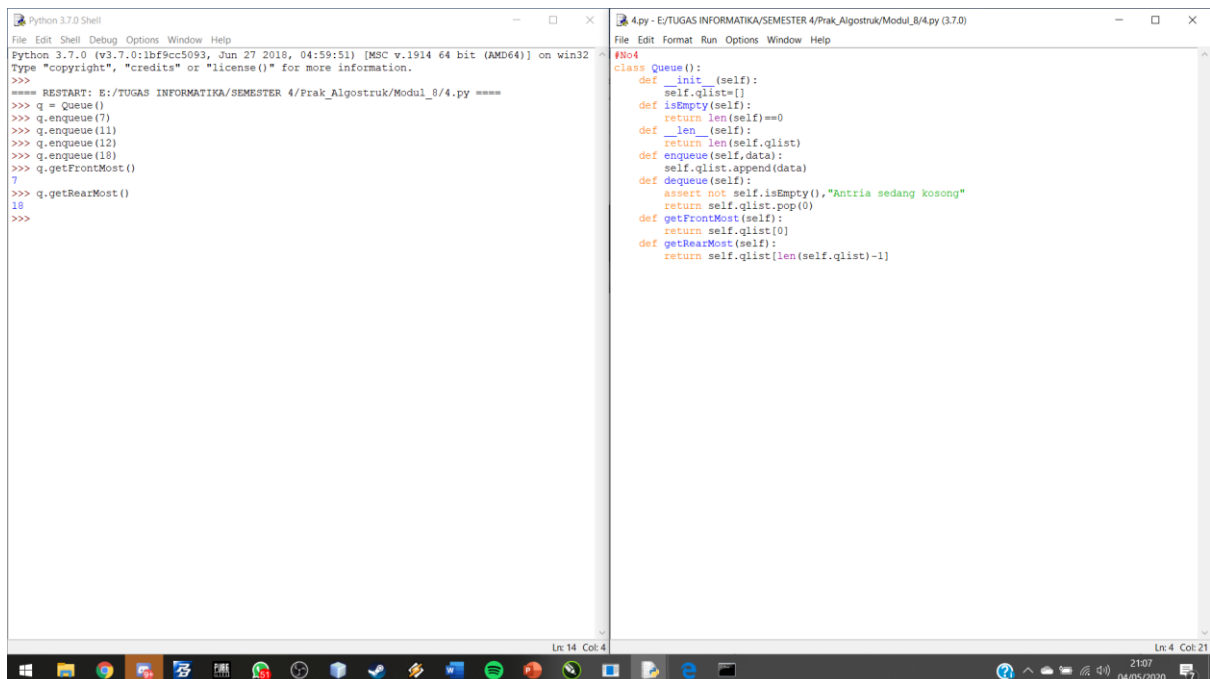
The screenshot shows a Python 3.7.0 Shell window on the left and a Python script editor on the right. The shell window displays the execution of a program that pushes numbers 0 to 15 onto a stack and then pops them. The script editor shows the implementation of a Stack class with methods: `__init__`, `isEmpty`, `len`, `peek`, `pop`, and `push`. The stack is implemented as a list.

```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (tags/v3.7.0:1bf9cc5093, Jun 27 2019, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:/TUGAS INFORMATIKA/SEMESTER 4/Prak_Algostruk/Modul_8/3.py =====
>>> nilai = Stack()
>>> for i in range(16):
>>>     if i % 3 == 0:
>>>         nilai.push(i)
>>>     elif i % 4 == 0:
>>>         nilai.pop()
>>>
>>>
3
6
>>> nilai.pop()
15
>>> nilai.pop()
12
>>> nilai.pop()
9
>>> nilai.pop()
0
>>>
```

```
3.py - E:/TUGAS INFORMATIKA/SEMESTER 4/Prak_Algostruk/Modul_8/3.py (3.7.0)
File Edit Format Run Options Window Help
#No3
class Stack():
    def __init__(self):
        self.items = []
    def isEmpty(self):
        return len(self) == 0
    def __len__(self):
        return len(self.items)
    def peek(self):
        assert not self.isEmpty()
        return self.items[-1]
    def pop(self):
        assert not self.isEmpty()
        return self.items.pop()
    def push(self, data):
        self.items.append(data)

nilai = Stack()
for i in range(16):
    if i % 3 == 0:
        nilai.push(i)
    elif i % 4 == 0:
        nilai.pop()
```

4. Tulis dua metode berikut ke class Query dan class PriorityQueue di atas
- Metode untuk mengetahui item yang paling depan tanpa menghapusnya



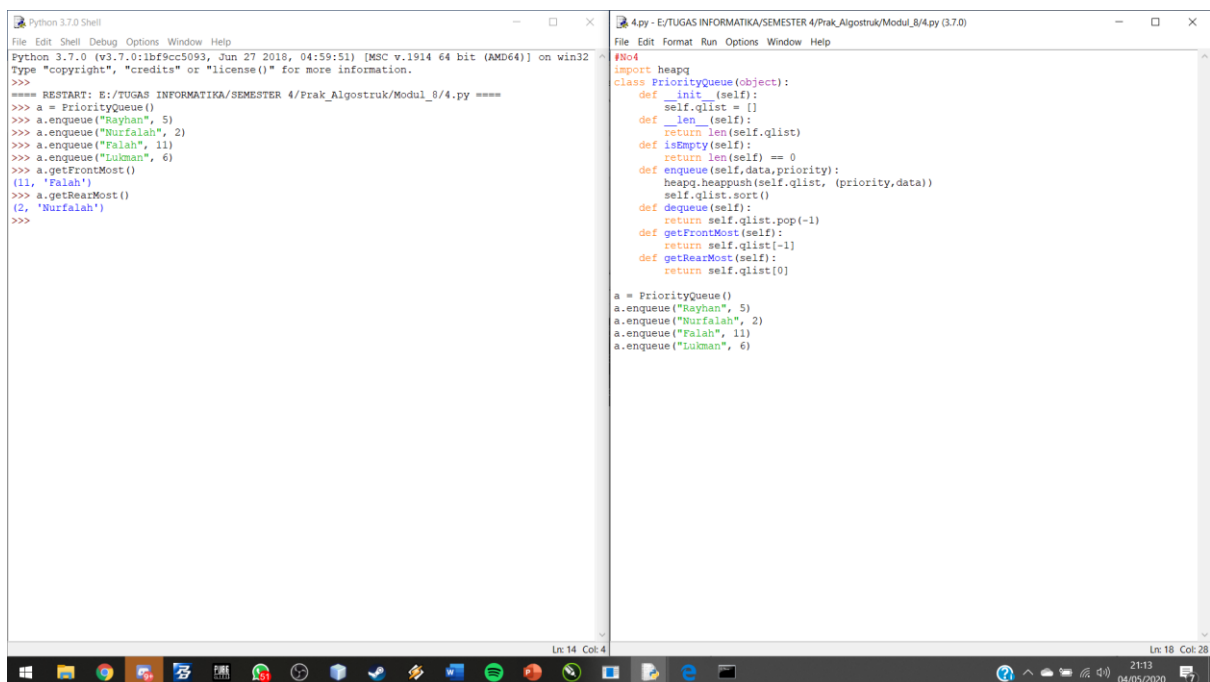
The screenshot shows two windows. The left window is a Python 3.7.0 Shell with the following code:

```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:/TUGAS INFORMATIKA/SEMESTER 4/Prak_Algostruk/Modul_8/4.py =====
>>> q = Queue()
>>> q.enqueue(7)
>>> q.enqueue(11)
>>> q.enqueue(12)
>>> q.enqueue(18)
>>> q.getFrontMost()
7
>>> q.getRearMost()
18
>>>
```

The right window is a Python file editor showing the implementation of the Queue class:

```
4.py - E:/TUGAS INFORMATIKA/SEMESTER 4/Prak_Algostruk/Modul_8/4.py (3.7.0)
File Edit Format Run Options Window Help
#No4
class Queue():
    def __init__(self):
        self.qlist=[]
    def isEmpty(self):
        return len(self)==0
    def __len__(self):
        return len(self.qlist)
    def enqueue(self,data):
        self.qlist.append(data)
    def dequeue(self):
        assert not self.isEmpty(),"Antrian sedang kosong"
        return self.qlist.pop(0)
    def getFrontMost(self):
        return self.qlist[0]
    def getRearMost(self):
        return self.qlist[len(self.qlist)-1]
```

- Metode untuk mengetahui item yang paling belakang tanpa menghapusnya



The screenshot shows two windows. The left window is a Python 3.7.0 Shell with the following code:

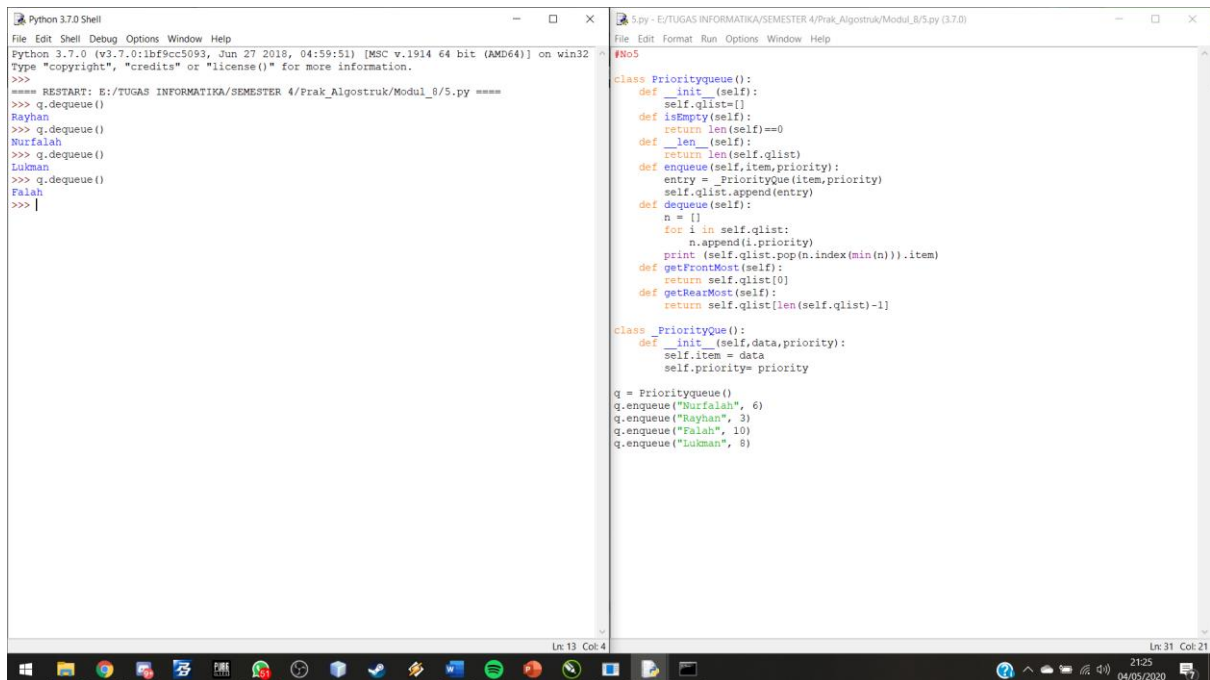
```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:/TUGAS INFORMATIKA/SEMESTER 4/Prak_Algostruk/Modul_8/4.py =====
>>> a = PriorityQueue()
>>> a.enqueue("Rayhan", 5)
>>> a.enqueue("Nurfalah", 2)
>>> a.enqueue("Falah", 11)
>>> a.enqueue("Lukman", 6)
>>> a.getFrontMost()
(11, 'Falah')
>>> a.getRearMost()
(2, 'Nurfalah')
>>>
```

The right window is a Python file editor showing the implementation of the PriorityQueue class:

```
4.py - E:/TUGAS INFORMATIKA/SEMESTER 4/Prak_Algostruk/Modul_8/4.py (3.7.0)
File Edit Format Run Options Window Help
#No4
import heapq
class PriorityQueue(object):
    def __init__(self):
        self.qlist = []
    def __len__(self):
        return len(self.qlist)
    def isEmpty(self):
        return len(self) == 0
    def enqueue(self,data,priority):
        heapq.heappush(self.qlist, (priority,data))
    def dequeue(self):
        self.qlist.sort()
        return self.qlist.pop(-1)
    def getFrontMost(self):
        return self.qlist[-1]
    def getRearMost(self):
        return self.qlist[0]

a = PriorityQueue()
a.enqueue("Rayhan", 5)
a.enqueue("Nurfalah", 2)
a.enqueue("Falah", 11)
a.enqueue("Lukman", 6)
```

5. Pada class PriorityQueue di atas, metode dequeue() belum diimplementasikan. Tulislah metode dequeue() ini dengan memperhatikan syarat-syarat seperti yang telah dicantumkan di halaman 81.



The screenshot shows a Python 3.7.0 Shell on the left and a Python script editor on the right. The shell displays the execution of a PriorityQueue class, showing the state of the queue after several enqueue and dequeue operations. The script editor shows the implementation of the PriorityQueue class, including methods for enqueue, dequeue, is_empty, len, getFrontMost, and getRearMost. The dequeue method is implemented to remove the element with the highest priority (lowest index) from the queue.

```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bbf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:/TUGAS INFORMATIKA/SEMESTER 4/Prak_Algostruk/Modul_8/5.py =====
>>> q.dequeue()
>>>
Rayhan
>>> q.dequeue()
Murfalah
>>> q.dequeue()
Lukman
>>> q.dequeue()
Falah
>>> |

5.py - E:/TUGAS INFORMATIKA/SEMESTER 4/Prak_Algostruk/Modul_8/5.py (3.7.0)
File Edit Format Run Options Window Help
#No5

class PriorityQueue():
    def __init__(self):
        self.qlist=[]
    def isEmpty(self):
        return len(self)==0
    def __len__(self):
        return len(self.qlist)
    def enqueue(self,item,priority):
        entry = _PriorityQue(item,priority)
        self.qlist.append(entry)
    def dequeue(self):
        n = []
        for i in self.qlist:
            n.append(i.priority)
        print (self.qlist.pop(n.index(min(n))).item)
    def getFrontMost(self):
        return self.qlist[0]
    def getRearMost(self):
        return self.qlist[len(self.qlist)-1]

class _PriorityQue():
    def __init__(self,data,priority):
        self.item = data
        self.priority= priority

q = PriorityQueue()
q.enqueue("Murfalah", 6)
q.enqueue("Rayhan", 3)
q.enqueue("Falah", 10)
q.enqueue("Lukman", 8)
```