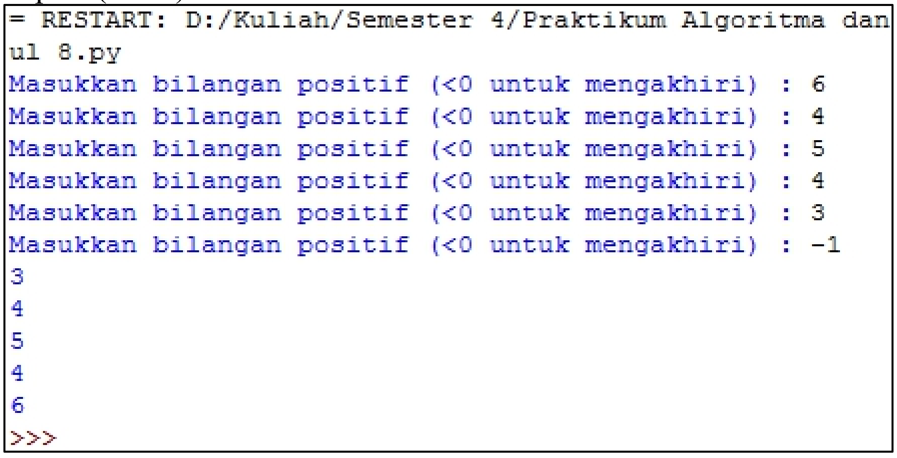
LAPORAN PRAKTIKUM ALGORITMA DAN STRUKTUR DATA MODUL 8 ”STACKS AND QUEUES”

Oleh:

NAMA	**: Daffa Putra Alwansyah**
NIM	**: L200190031**
KELAS	**: B**
PRODI	**: Informatika**

Fakultas Komunikasi dan Informatika
Universitas Muhammadiyah Surakarta

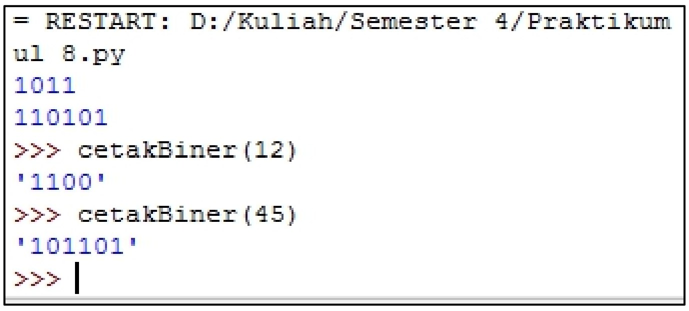
```
#=============Latihan===================
##=========Latihan 8.3=========
class Stack:
    def __init__(self):
        self.items = []

    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)

class StackLL:
    def __init__(self):
        self.top = None
        self.size = 0

    def isEmpty(self):
        return self.top is None

    def __len__(self):
        return self.size

    def peek(self):
        assert not self.isEmpty()
        return self.top.item

    def pop(self):
        assert not self.isEmpty()
        node = self.top
        self.top = self.top.next
        self.size -= 1
        return node.item

    def push(self):
        self.top = _StackNode(data, self.top)
        self.size += 1

class _StackNode:
    def __init__(self, data, link):
        self.item = data
        self.next = link

PROMPT = "Masukkan bilangan positif (<0 untuk mengakhiri) : "
myStack = Stack()
value = int(input(PROMPT))
while value >= 0:
    myStack.push(value)
    value = int(input(PROMPT))
while not myStack.isEmpty():
    value = myStack.pop()
    print(value)
```



```
= RESTART: D:/Kuliah/Semester 4/Praktikum Algoritma dan
ul 8.py
Masukkan bilangan positif (<0 untuk mengakhiri) : 6
Masukkan bilangan positif (<0 untuk mengakhiri) : 4
Masukkan bilangan positif (<0 untuk mengakhiri) : 5
Masukkan bilangan positif (<0 untuk mengakhiri) : 4
Masukkan bilangan positif (<0 untuk mengakhiri) : 3
Masukkan bilangan positif (<0 untuk mengakhiri) : -1
3
4
5
4
6
>>>
```

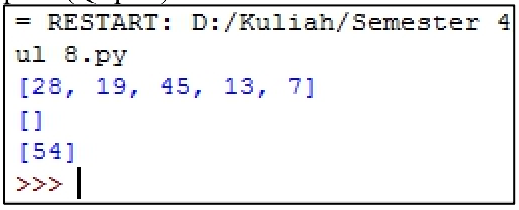
##=========Latihan 8.4=========

```
def cetakBiner(d):
    f = Stack()
    if d==0: f.push(0);
    while d !=0:
        sisa = d%2
        d = d//2
        f.push(sisa)
    st = ""
    for i in range(len(f)):
        st = st + str(f.pop())
    return st

print(cetakBiner(11))
print(cetakBiner(53))
```



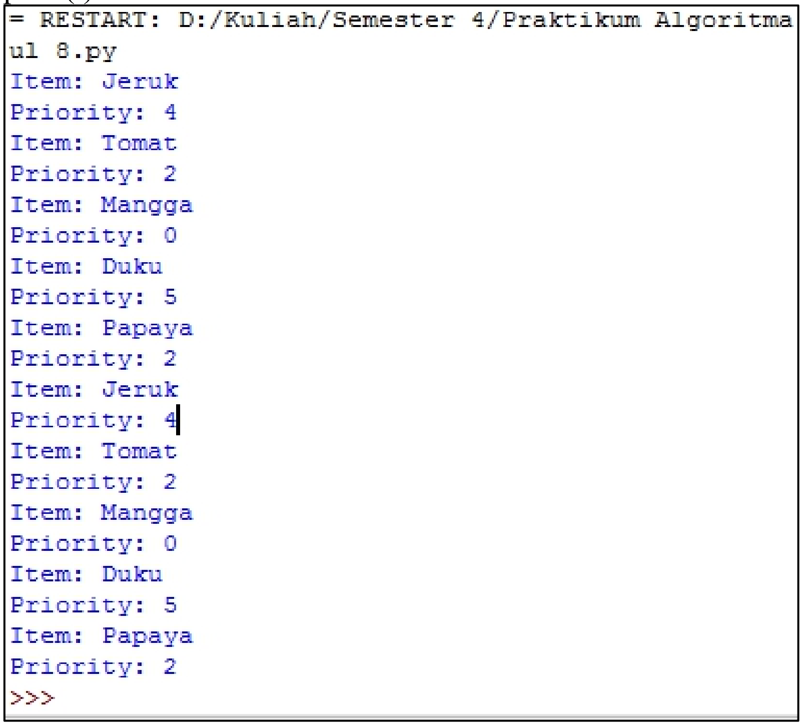
```
= RESTART: D:/Kuliah/Semester 4/Praktikum
ul 8.py
1011
110101
>>> cetakBiner(12)
'1100'
>>> cetakBiner(45)
'101101'
>>> 
```

```
##=========Latihan 8.6=========
class Queue(object):
    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)

Q = Queue()
Q.enqueue(28)
Q.enqueue(19)
Q.enqueue(45)
Q.enqueue(13)
Q.enqueue(7)
print(Q.qlist)
Q.dequeue()
Q.dequeue()
Q.dequeue()
Q.dequeue()
Q.dequeue()
print(Q.qlist)
Q.enqueue(98)
Q.enqueue(54)
Q.dequeue()
print(Q.qlist)
```



```
= RESTART: D:/Kuliah/Semester 4
ul 8.py
[28, 19, 45, 13, 7]
[]
[54]
>>> 
```

```
##=========Latihan 8.7=========
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):
        entry = _PriorityQEntry(data, priority)
        self.qlist.append(entry)

    def dequeue(self):
        pass


class _PriorityQEntry(object):

    def __init__(self, data, priority):
         self.item = data
         self.priority = priority
    def __str__(self):
        return 'Item: {}\nPriority: {}'.format(self.item, self.priority)

S = PriorityQueue()
S.enqueue('Jeruk', 4)
S.enqueue('Tomat', 2)
S.enqueue('Mangga', 0)
S.enqueue('Duku', 5)
S.enqueue('Papaya', 2)
for i in S.qlist:
    print(i)
S.dequeue()
S.dequeue()
S.dequeue()
for i in S.qlist:
    print(i)
```

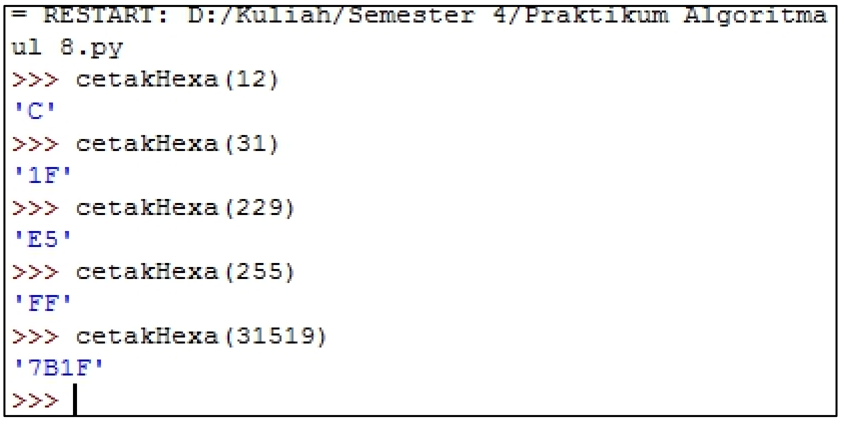


```
= RESTART: D:/Kuliah/Semester 4/Praktikum Algoritma
ul 8.py
Item: Jeruk
Priority: 4
Item: Tomat
Priority: 2
Item: Mangga
Priority: 0
Item: Duku
Priority: 5
Item: Papaya
Priority: 2
Item: Jeruk
Priority: 4
Item: Tomat
Priority: 2
Item: Mangga
Priority: 0
Item: Duku
Priority: 5
Item: Papaya
Priority: 2
>>>
```

##============Soal Mahasiswa============

##=========Nomor 1=========

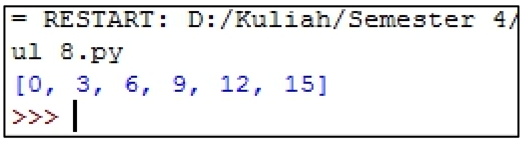
```
class Stack(object):
    def __init__(self):
        self.items = []

    def isEmpty(self):
        return len(self) == 0

    def __len__(self):
        return len(self.items)

    def peek(self):
        assert not self.isEmpty(), "Tidak bisa diintip. Stack kosong"
        return self.items[-1]

    def pop(self):
        assert not self.isEmpty(), "Tidak bisa dipop dari Stack kosong"
        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:
            sisa = d%16
            d = d//16
            if sisa == 10:
                sisa = "A"
            elif sisa == 11:
                sisa = "B"
            elif sisa == 12:
                sisa = "C"
            elif sisa == 13:
                sisa = "D"
            elif sisa == 14:
                sisa = "E"
            elif sisa == 15:
                sisa = "F"
            f.push(sisa)
        st = ""
        for i in range (len(f)):
            st = st + str(f.pop())
        return st
```



```
= RESTART: D:/Kuliah/Semester 4/Praktikum Algoritma
ul 8.py
>>> cetakHexa(12)
'C'
>>> cetakHexa(31)
'1F'
>>> cetakHexa(229)
'E5'
>>> cetakHexa(255)
'FF'
>>> cetakHexa(31519)
'7B1F'
>>> |
```

##=========Nomor 2=========

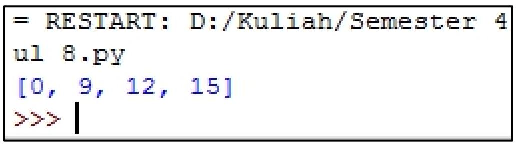
```
class Stack(object):
    def __init__(self):
        self.items = []

    def isEmpty(self):
        return len(self) == 0

    def __len__(self):
        return len(self.items)

    def peek(self):
        assert not self.isEmpty(), "Tidak bisa diintip. Stack kosong"
        return self.items[-1]

    def pop(self):
        assert not self.isEmpty(), "Tidak bisa dipop dari Stack kosong"
        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)
print(nilai.items)
```



```
= RESTART: D:/Kuliah/Semester 4/
ul 8.py
[0, 3, 6, 9, 12, 15]
>>> |
```

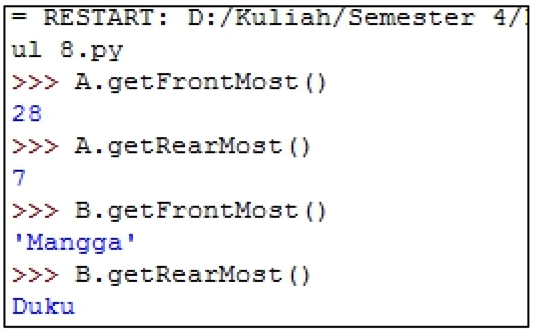
##=========Nomor 3=========

```
class Stack(object):
    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(), "Tidak bisa diintip. Stack kosong"
        return self.items[-1]

    def pop(self):
        assert not self.isEmpty(), "Tidak bisa dipop dari Stack kosong"
        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()
print(nilai.items)
```



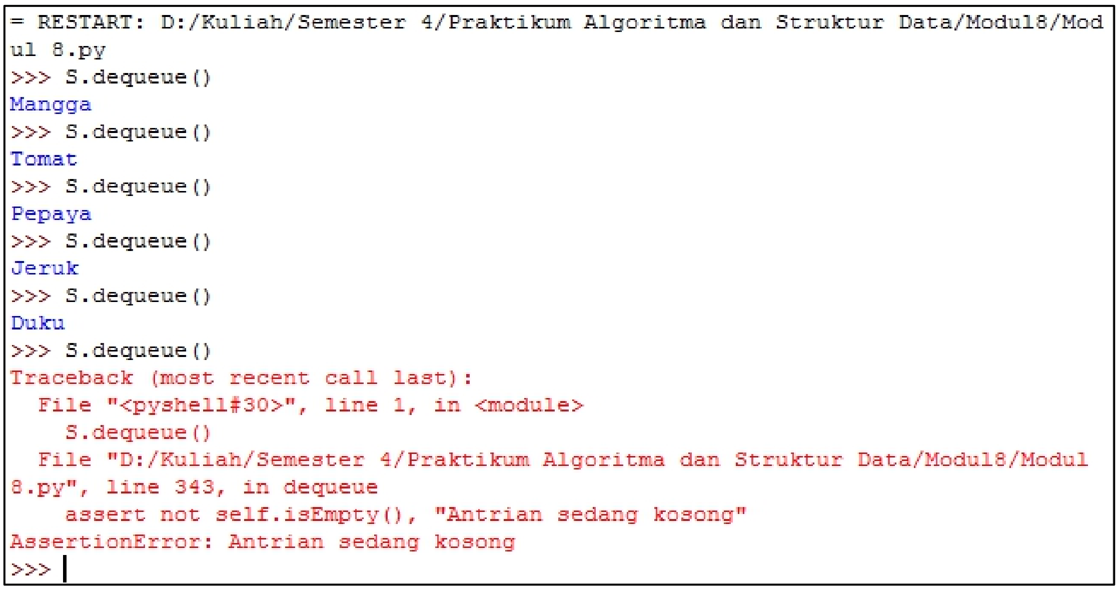
```
= RESTART: D:/Kuliah/Semester 4
ul 8.py
[0, 9, 12, 15]
>>>
```

```
##=========Nomor 4=========
class Queue(object):
    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[-1]

class PriorityQueue(object):
    def __init__(self):
        self.qlist = []
    def isEmpty(self):
        return len(self) == 0
    def __len__(self):
        return len(self.qlist)
    def enqueue(self, data, priority):
        entry = _PriorityQEntry(data, priority)
        self.qlist.append(entry)

    def getFrontMost(self):
        x = 0
        while self.qlist[x].priority != 0:
            x+=1
        return self.qlist[x].item

    def getRearMost(self):
        a = []
        for i in self.qlist:
            a.append(i.priority)
        print (self.qlist[a.index(max(a))].item)

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

A = Queue()
A.enqueue(28)
A.enqueue(19)
A.enqueue(45)
A.enqueue(13)
A.enqueue(7)

B = PriorityQueue()
B.enqueue("Jeruk", 4)
B.enqueue("Tomat", 2)
B.enqueue("Mangga", 0)
B.enqueue("Duku", 5)
B.enqueue("Pepaya", 2)
```



```
= RESTART: D:/Kuliah/Semester 4/
ul 8.py
>>> A.getFrontMost()
28
>>> A.getRearMost()
7
>>> B.getFrontMost()
'Mangga'
>>> B.getRearMost()
Duku
```

```
##=========Nomor 5=========
class PriorityQueue(object):
    def __init__(self):
        self.qlist = []

    def isEmpty(self):
        return len(self) == 0

    def __len__(self):
        return len(self.qlist)

    def enqueue(self, data, priority):
        entry = _PriorityQEntry(data, priority)
        self.qlist.append(entry)

    def dequeue(self):
        assert not self.isEmpty(), "Antrian sedang kosong"
        a = []
        for i in self.qlist:
            a.append(i.priority)
        print (self.qlist.pop(a.index(min(a))).item)

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

S = PriorityQueue()
S.enqueue("Jeruk", 4)
S.enqueue("Tomat", 2)
S.enqueue("Mangga", 0)
S.enqueue("Duku", 4)
S.enqueue("Pepaya", 2)
```



```
= RESTART: D:/Kuliah/Semester 4/Praktikum Algoritma dan Struktur Data/Modul8/Mod
ul 8.py
>>> S.dequeue()
Mangga
>>> S.dequeue()
Tomat
>>> S.dequeue()
Pepaya
>>> S.dequeue()
Jeruk
>>> S.dequeue()
Duku
>>> S.dequeue()
Traceback (most recent call last):
  File "<pyshell#30>", line 1, in <module>
    S.dequeue()
  File "D:/Kuliah/Semester 4/Praktikum Algoritma dan Struktur Data/Modul8/Modul
8.py", line 343, in dequeue
    assert not self.isEmpty(), "Antrian sedang kosong"
AssertionError: Antrian sedang kosong
>>> 
```