

PRAKTIKUM
ALGORITMA DAN STRUKTUR DATA
(Algorithm and Data Structure)

LAPORAN TUGAS
MODUL 8



Nama : Shafa Bani Saputra

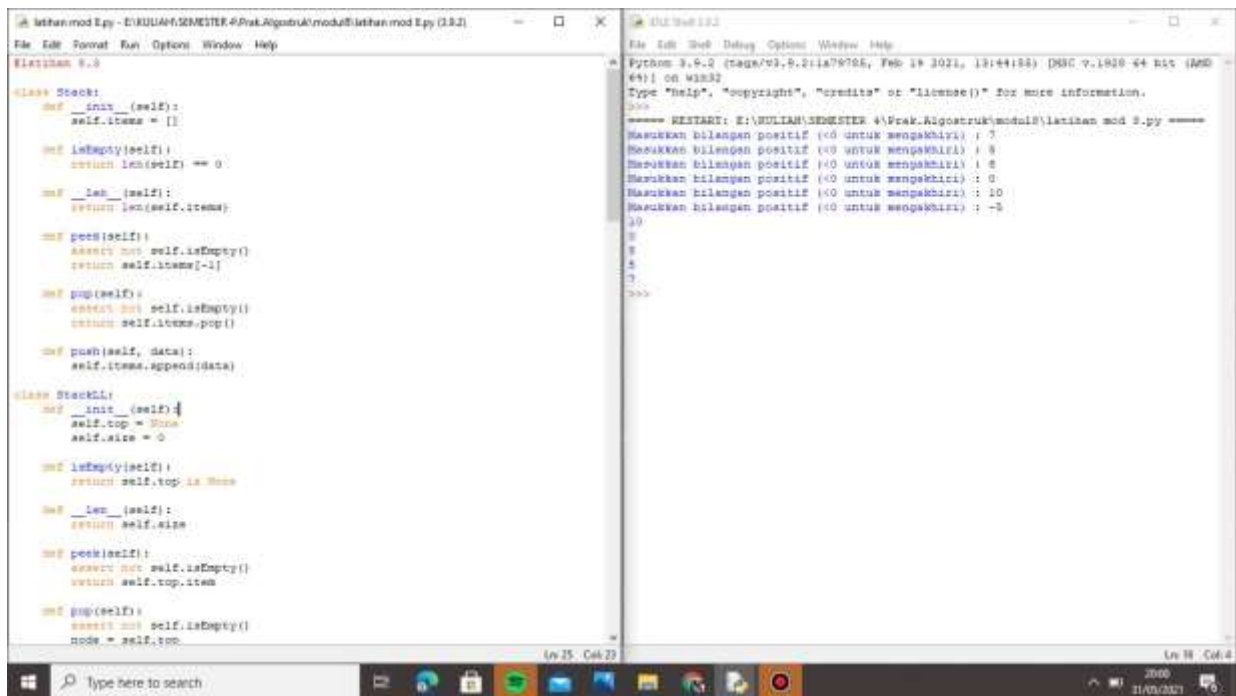
NIM : L200190151

Kelas : G

PROGRAM STUDI INFORMATIKA
FAKULTAS KOMUNIKASI DAN INFORMATIKA
UNIVERSITAS MUHAMMADIYAH
SURAKARTA

LATIHAN

8.3



```
latihan mod 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 StackML:
    def __init__(self):
        self.top = None
        self.size = 0

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

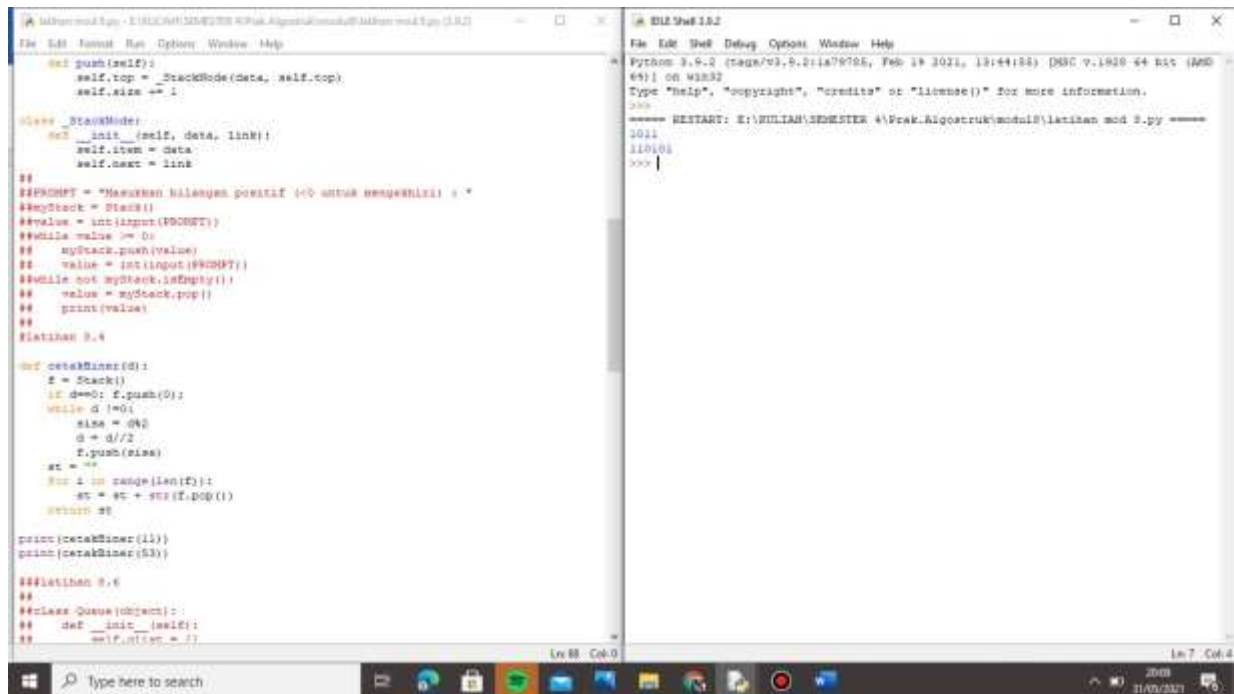
    def __len__(self):
        return self.size

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

    def pop(self):
        assert not self.isEmpty()
        node = self.top

latihan mod 8.2
Python 3.9.2 (tags/v3.9.2:1a79705, Feb 19 2021, 13:44:58) [MSI 7.1920 64 bit (AMD
64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:\NULIAH\SEMESTER 4\Prak.Algostruk\modul8\latihan mod 8.py =====
Masukkan bilangan positif (<0 untuk mengakhiri) : 7
Masukkan bilangan positif (<0 untuk mengakhiri) : 8
Masukkan bilangan positif (<0 untuk mengakhiri) : 8
Masukkan bilangan positif (<0 untuk mengakhiri) : 0
Masukkan bilangan positif (<0 untuk mengakhiri) : 10
Masukkan bilangan positif (<0 untuk mengakhiri) : -5
>>>
```

8.4



```
latihan mod 8.4
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)

latihan 8.4
def cetakDiner(d):
    s = Stack()
    if d==0: s.push(0)
    while d != 0:
        size = d%2
        d = d//2
        s.push(size)
    st = ""
    for i in range(len(s)-1):
        st = st + str(s.pop())
    return st

print(cetakDiner(1))
print(cetakDiner(5))

latihan 8.6
class Queue:
    def __init__(self):
        self.items = []

latihan mod 8.2
Python 3.9.2 (tags/v3.9.2:1a79705, Feb 19 2021, 13:44:58) [MSI 7.1920 64 bit (AMD
64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:\NULIAH\SEMESTER 4\Prak.Algostruk\modul8\latihan mod 8.py =====
1011
110101
>>>
```

8.6

The screenshot shows an IDE with two windows. The left window displays a Python script for a Queue class, and the right window shows the output of the script.

```

##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(20)
Q.enqueue(15)
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(99)
Q.enqueue(54)
Q.dequeue()
print(Q.qlist)

##latihan 8.7
##
##class PriorityQueue(object):

```

The right window shows the output of the script:

```

Python 3.9.2 (tags/v3.9.2:1a79780, Feb 19 2021, 13:44:55) [AMD64] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:\KULIAH\SEMESTER 4\Prak.Algostruk\modul8\latihan mod 8.py =====
[20, 15, 45, 13, 7]
[]
[54]
>>>

```

8.7

The screenshot shows an IDE with two windows. The left window displays a Python script for a PriorityQueue class, and the right window shows the output of the script.

```

##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)

P = PriorityQueue()
P.enqueue('Jeruk', 4)
P.enqueue('Tomat', 2)
P.enqueue('Mangga', 0)
P.enqueue('Duku', 5)
P.enqueue('Pepaya', 3)
for i in P.qlist:
    print(i)
P.dequeue()
P.dequeue()
P.dequeue()

```

The right window shows the output of the script:

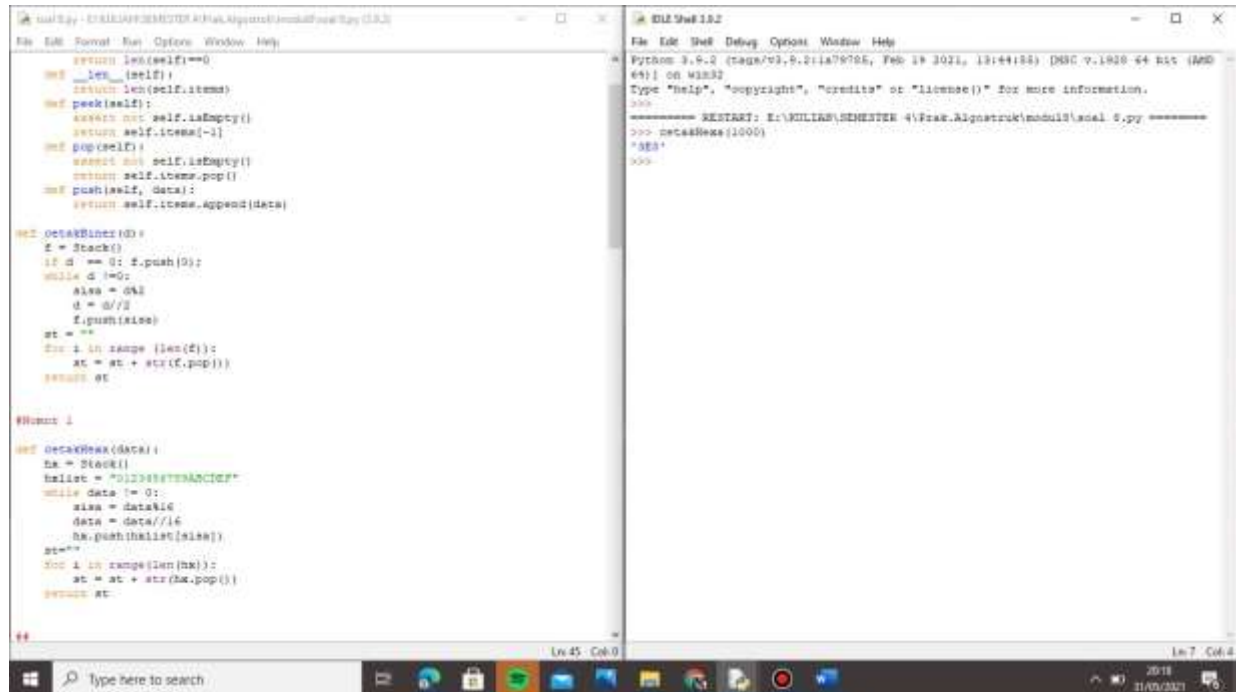
```

Python 3.9.2 (tags/v3.9.2:1a79780, Feb 19 2021, 13:44:55) [AMD64] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:\KULIAH\SEMESTER 4\Prak.Algostruk\modul8\latihan mod 8.py =====
Item: Jeruk
Priority: 4
Item: Tomat
Priority: 2
Item: Mangga
Priority: 0
Item: Duku
Priority: 5
Item: Pepaya
Priority: 3
Item: Jeruk
Priority: 4
Item: Tomat
Priority: 2
Item: Mangga
Priority: 0
Item: Duku
Priority: 5
Item: Pepaya
Priority: 3
>>>

```

TUGAS

1.



```
File Edit Format Run Options Window Help
class Stack:
    def __init__(self):
        self.items = []
    def push(self, data):
        self.items.append(data)
    def pop(self):
        if not self.isEmpty():
            return self.items.pop()
        else:
            return None
    def isEmpty(self):
        return len(self.items) == 0
    def peek(self):
        if not self.isEmpty():
            return self.items[-1]
        else:
            return None

def otakarBiner10():
    s = Stack()
    if s.isEmpty():
        s.push(0)
    while s != 0:
        sisa = s % 2
        s = s // 2
        s.push(sisa)
    st = ""
    for i in range(len(s)):
        st = str(s.pop()) + st
    return st

#Hukum 1
def otakarHexa(data):
    s = Stack()
    hexlist = "0123456789ABCDEF"
    while data != 0:
        sisa = data % 16
        data = data // 16
        s.push(hexlist[sisa])
    st = ""
    for i in range(len(s)):
        st = str(s.pop()) + st
    return st

if __name__ == '__main__':
    otakarBiner10()
    otakarHexa(1000)
```

```
Python 3.9.2 (tags/v3.9.2:1a79705, Feb 19 2021, 13:44:34) [AMD64] on win32
Type "help()", "copyright()", "credits()" or "license()" for more information.
>>> otakarHexa(1000)
800
>>>
```

Ln 45 Col 0

Ln 7 Col 4

Type here to search

2021
11/03/2021

2.

```

File Edit Format Run Options Window Help
## Stack Implementation
def push(stack, value):
    stack.append(value)

def pop(stack):
    if len(stack) > 0:
        return stack.pop()
    else:
        return None

# Test Stack
stack = []
push(stack, 10)
push(stack, 20)
push(stack, 30)
print(stack)  # [10, 20, 30]
pop(stack)    # 30
pop(stack)    # 20
pop(stack)    # 10
pop(stack)    # None

## Queue Implementation
class Queue:
    def __init__(self):
        self.queue = []

    def is_empty(self):
        return len(self.queue) == 0

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

    def enqueue(self, data):
        self.queue.append(data)

    def dequeue(self):
        if not self.is_empty():
            return self.queue.pop(0)
        else:
            return None

# Test Queue
q = Queue()
q.enqueue(10)
q.enqueue(20)
q.enqueue(30)
print(q.queue)  # [10, 20, 30]
q.dequeue()     # 10
q.dequeue()     # 20
q.dequeue()     # 30
q.dequeue()     # None
  
```

Python 3.9.2 [tags/v3.9.2:107952, Feb 19 2021, 13:44:55] [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: Shell: C:\Python39\Scripts\python.exe =====
[10, 20, 30]
30
20
10
None
[10, 20, 30]
10
20
30
None
>>>

3.

```

File Edit Format Run Options Window Help
## Stack Implementation
def push(stack, value):
    stack.append(value)

def pop(stack):
    if len(stack) > 0:
        return stack.pop()
    else:
        return None

# Test Stack
stack = []
push(stack, 10)
push(stack, 20)
push(stack, 30)
print(stack)  # [10, 20, 30]
pop(stack)    # 30
pop(stack)    # 20
pop(stack)    # 10
pop(stack)    # None

## Queue Implementation
class Queue:
    def __init__(self):
        self.queue = []

    def is_empty(self):
        return len(self.queue) == 0

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

    def enqueue(self, data):
        self.queue.append(data)

    def dequeue(self):
        if not self.is_empty():
            return self.queue.pop(0)
        else:
            return None

# Test Queue
q = Queue()
q.enqueue(10)
q.enqueue(20)
q.enqueue(30)
print(q.queue)  # [10, 20, 30]
q.dequeue()     # 10
q.dequeue()     # 20
q.dequeue()     # 30
q.dequeue()     # None
  
```

Python 3.9.2 [tags/v3.9.2:107952, Feb 19 2021, 13:44:55] [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: Shell: C:\Python39\Scripts\python.exe =====
[10, 20, 30]
30
20
10
None
[10, 20, 30]
10
20
30
None
>>>

4. Queue

The screenshot shows a code editor on the left and a terminal window on the right. The code editor contains the implementation of a Queue class using a list. The terminal window shows the output of the program.

```
##Homework 4 Queue
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()
        return self.qlist.pop(0)
    def getFront(self):
        return self.qlist[-1]
    def getRear(self):
        return self.qlist[0]

a = Queue()
a.enqueue('aa')
a.enqueue('bb')
a.enqueue('cc')

print(a.qlist)
a.dequeue()
print(a.qlist)

print(a.getFront())
print(a.getRear())

##Homework 4 PriorQueue
##import heapq
##class PriorQueue(object):
##    def __init__(self):
##        self.qlist = []
##    def __len__(self):
##        return len(self.qlist)
##    def enqueue(self, data, priority):
##        heapq.heappush(self.qlist, (priority, data))
##    def dequeue(self):
##        return self.qlist.pop(0)
```

The terminal window shows the output of the program:

```
Python 3.9.2 [tags/v3.9.2:1a79795, Feb 19 2021, 13:44:55] [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:\KULIAH\SEMESTER 4\Frak.Algostrak\modul9\asal 0.py =====
['aa', 'bb', 'cc']
['bb', 'cc']
cc
bb
>>>
```

4. PriorQueue

The screenshot shows a code editor on the left and a terminal window on the right. The code editor contains the implementation of a PriorQueue class using a heap. The terminal window shows the output of the program.

```
##Homework 4 PriorQueue
##import heapq
##class PriorQueue(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):
##        return self.qlist.pop(0)

a = PriorQueue()
a.enqueue('aa', 1)
a.enqueue('bb', 2)
a.enqueue('cc', 2)
print(a.qlist)

##Homework 5
import heapq
class PriorQueue(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):
        return self.qlist.pop(0)

a = PriorQueue()
a.enqueue('aa', 1)
a.enqueue('bb', 2)
a.enqueue('cc', 2)

print(a.qlist)
a.dequeue()
print(a.qlist)
```

The terminal window shows the output of the program:

```
Python 3.9.2 [tags/v3.9.2:1a79795, Feb 19 2021, 13:44:55] [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:\KULIAH\SEMESTER 4\Frak.Algostrak\modul9\asal 0.py =====
[(1, 'aa'), (2, 'cc'), (2, 'bb')]
[(1, 'aa'), (2, 'cc')]
>>>
```

