NAMA : KEVIN AVICENNA WIDIARTO
NIM : L200200183
Modul : 8

Praktikum Algoritma Struktur Data

MODUL 8

No 1

File Edit Format Run Options Window Help

```python
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()
        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(b):
    S = Stack() #menyimpan class stack ke var S

    if b == 0:
        S.push(0)

    while b !=0:
        sisa = b%16
        b = b//16
        S.push(sisa)
        hexa = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 'A', 'B', 'C', 'D', 'E', 'F']

    hasil = ""

    for i in range (len(S)):
        hasil += str(hexa[S.pop()])

    return hasil
```

File Edit Shell Debug

```
========= RESTART:
>>> cetakHexa(32)
'20'
>>> cetakHexa(70)
'46'
>>> cetakHexa(120)
'78'
>>> cetakHexa(180)
'B4'
>>>
```

No 2

```python
from no1 import Stack

S1 = Stack()
for i in range(16):
    if i % 3 == 0:
        S1.push(i)

print(S1.items)
```

IDLE Shell 3.10.4

File Edit Shell Debug Options Window

```
[0, 3, 6, 9, 12, 15]
>>>
```

No 3

```python
from no1 import Stack

S2 = Stack()
for i in range (16) :
    if i % 3 == 0:
        S2.push(i)
    elif i % 4 == 0:
        S2.pop()

print(S2.items)
```

File Edit Shell Debug Options Windo

```
[0, 9, 12, 15]
>>>
```

No 4

```python
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)

    # FUNGSI untuk mengetahui yg paling dpn tanpa menghapus
    # dan mengetahui item yg paling belakang tanpa menghapus
    def getFrontMost(self):
        assert not self.isEmpty(), "Antrian sedang kosong."
        return self.qlist[0]
    def getRearMost(self):
        assert not self.isEmpty(), "Antrian sedang kosong."
        return self.qlist[-1]

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

print (c.qlist)
print ("Item terdepan : " , c.getFrontMost())
print ("Item terbelakang : " , c.getRearMost())

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)
```

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

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

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


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

print (d.qlist)
print ("Item paling depan : " , d.getFrontMost())
print ("Item paling belakang : " , d.getRearMost())
```

No 5

```python
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):
        tmp = []
        for i in self.qlist:
            tmp.append(i)
        s = 0
        for i in range(1, len(self.qlist)):
            if tmp[i].priority < tmp[s].priority:
                s = i
        hasil = self.qlist.pop(s)
        return hasil.item


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

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

print (d.dequeue())
print (d.dequeue())
print (d.dequeue())
print (d.dequeue())
print (d.dequeue())
```