Datenstrukturen (Kapitel 2)

Prüfungsvorbereitung

Aufgabe 1

Implementiere auf der Grundlage des folgenden Klassengerüsts die unvollständig definierten Methoden.

```
1  class Stack:
2
3      def __init__(self):
4          self.data = []
5
6      def push(self, item):
7          ...
8
9      def pop(self):
10         ...
11
12     def peek(self):
13         ...
14
15     def size(self):
16         ...
17
```

Aufgabe 1

```
1  class Stack:
2
3      def __init__(self):
4          self.data = []
5
6      def push(self, item):
7          self.data.append(item)
8
9      def pop(self):
10         return self.data.pop()
11
12     def peek(self):
13         return self.data[-1]
14
15     def size(self):
16         return len(self.data)
17
18     def isEmpty(self):
19         return self.data == []
```

Aufgabe 2

Gib den Zustand des Stacks `s` am Ende des folgenden Python-Programms an (ältestes Element unten).

```
1  from stack import Stack
2
3  s = Stack()
4  s.push(7)
5  s.push(8)
6  s.push(4)
7  s.pop()
8  s.pop()
9  s.push(3)
10 s.push(1)
11 s.pop()
```

Aufgabe 2

3

7

Aufgabe 3

Wofür steht die Kurzformel LIFO?

Aufgabe 3

Für die Funktionsweise eines Stacks: *Last In – First Out*

Aufgabe 4

Zähle drei mögliche Anwendungen des abstrakten Datentyps Stack auf.

Aufgabe 4

- Undo/Redo-Funktionen von Anwenderprogrammen (Browser-History)
- Auswertung von Ausdrücken (z. B. UPN) und Parsen von Syntax (z. B. Ausdrücke auf korrekte Verschachtelung prüfen)
- Verwaltung des Arbeitsspeichers von Computern
- Als Datenstruktur für Algorithmen (z. B. Lösungsalgorithmen für Irrgärten)

Aufgabe 5

Angenommen ein Programm führt eine abwechselnde Folge von `push`- und `pop`-Operationen auf einem Stack aus. Die `push`-Operationen legen dabei die ganzen Zahlen von 0 bis 9 in dieser Reihenfolge auf den Stack ab. Die `pop`-Operationen geben den Rückgabewert aus. Welche der folgenden Sequenzen kann es nicht geben?

(a) `4 3 2 1 0 9 8 7 6 5`

(b) `4 6 8 7 5 3 2 9 0 1`

(c) `2 5 6 7 4 8 9 3 1 0`

(d) `4 3 2 1 0 5 6 7 8 9`

Aufgabe 5

(a) 4 3 2 1 0 9 8 7 6 5

Aufgabe 5

(a) 4 3 2 1 0 9 8 7 6 5

0

Aufgabe 5

(a)

```
4 3 2 1 0 9 8 7 6 5
    0   1
```

Aufgabe 5

(a)

```
4 3 2 1 0 9 8 7 6 5
    0   1   2
```

Aufgabe 5

(a)

4	3	2	1	0	9	8	7	6	5
	0		1		2		3		

Aufgabe 5

(a)

4	3	2	1	0	9	8	7	6	5
	0		1		2		3		4

Aufgabe 5

(a)

4	3	2	1	0	9	8	7	6	5
	0		1		2		3		~~4~~

4

Aufgabe 5

(a) 4 3 2 1 0 9 8 7 6 5

0 1 2 ~~3~~ ~~4~~

4 3

Aufgabe 5

(a) 4 3 2 1 0 9 8 7 6 5

0 1 ~~2~~ ~~3~~ ~~4~~

4 3 2

Aufgabe 5

(a) 4 3 2 1 0 9 8 7 6 5

0 ~~1~~ ~~2~~ ~~3~~ ~~4~~

4 3 2 1

Aufgabe 5

(a)

4	3	2	1	0	9	8	7	6	5
~~0~~	~~1~~	~~2~~	~~3~~	~~4~~					
4	3	2	1	0					

Aufgabe 5

(a) 4 3 2 1 0 9 8 7 6 5

~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5

4 3 2 1 0

Aufgabe 5

(a)

4	3	2	1	0	9	8	7	6	5
~~0~~	~~1~~	~~2~~	~~3~~	~~4~~	5	6			
4	3	2	1	0					

Aufgabe 5

(a) 4 3 2 1 0 9 8 7 6 5

~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5 6 7

4 3 2 1 0

Aufgabe 5

(a) 4 3 2 1 0 9 8 7 6 5

~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5 6 7 8

4 3 2 1 0

Aufgabe 5

(a) 4 3 2 1 0 9 8 7 6 5

~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5 6 7 8 9

4 3 2 1 0

Aufgabe 5

(a) 4 3 2 1 0 9 8 7 6 5

~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5 6 7 8 ~~9~~

4 3 2 1 0 9

Aufgabe 5

(a) 4 3 2 1 0 9 8 7 6 5

~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5 6 7 ~~8~~ ~~9~~

4 3 2 1 0 9 8

Aufgabe 5

(a) 4 3 2 1 0 9 8 7 6 5

~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5 6 ~~7~~ ~~8~~ ~~9~~

4 3 2 1 0 9 8 7

Aufgabe 5

(a) 4 3 2 1 0 9 8 7 6 5

~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5 ~~6~~ ~~7~~ ~~8~~ ~~9~~

4 3 2 1 0 9 8 7 6

Aufgabe 5

(a) 4 3 2 1 0 9 8 7 6 5

~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ ~~9~~

4 3 2 1 0 9 8 7 6 5

Aufgabe 5

(a) 4 3 2 1 0 9 8 7 6 5

~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ ~~9~~

4 3 2 1 0 9 8 7 6 5 True

(b) 4 6 8 7 5 3 2 9 0 1

(b) 4 6 8 7 5 3 2 9 0 1

0

(b) 4 6 8 7 5 3 2 9 0 1

0 1

(b) 4 6 8 7 5 3 2 9 0 1

0 1 2

(b) 4 6 8 7 5 3 2 9 0 1

0 1 2 3

(b)

4	6	8	7	5	3	2	9	0	1
0	1	2	3	4					

(b) 4 6 8 7 5 3 2 9 0 1

0 1 2 3 ~~4~~

4

(b) 4 6 8 7 5 3 2 9 0 1

0 1 2 3 ~~4~~ 5

4

(b) 4 6 8 7 5 3 2 9 0 1

0 1 2 3 ~~4~~ 5 6

4

(b) 4 6 8 7 5 3 2 9 0 1

0 1 2 3 ~~4~~ 5 ~~6~~

4 6

(b) 4 6 8 7 5 3 2 9 0 1

0 1 2 3 ~~4~~ 5 ~~6~~ 7

4 6

(b) 4 6 8 7 5 3 2 9 0 1

0 1 2 3 ~~4~~ 5 ~~6~~ 7 8

4 6

(b) 4 6 8 7 5 3 2 9 0 1

0 1 2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~

4 6 8

(b) 4 6 8 7 5 3 2 9 0 1

0 1 2 3 ~~4~~ 5 ~~6~~ ~~7~~ ~~8~~

4 6 8 7

(b) 4 6 8 7 5 3 2 9 0 1

0 1 2 3 ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~

4 6 8 7 5

(b) 4 6 8 7 5 3 2 9 0 1

0 1 2 ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~

4 6 8 7 5 3

(b) 4 6 8 7 5 3 2 9 0 1

0 1 ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~

4 6 8 7 5 3 2

(b) 4 6 8 7 5 3 2 9 0 1

0 1 ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ 9

4 6 8 7 5 3 2

(b) 4 6 8 7 5 3 2 9 0 1

0 1 ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ ~~9~~

4 6 8 7 5 3 2 9

(b) 4 6 8 7 5 3 2 9 0 1

0 ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ ~~9~~

4 6 8 7 5 3 2 9 1

(b) 4 6 8 7 5 3 2 9 0 1

~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ ~~9~~

4 6 8 7 5 3 2 9 1 0

(b) 4 6 8 7 5 3 2 9 0 1

~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ ~~9~~

4 6 8 7 5 3 2 9 1 0 False

(c) 2 5 6 7 4 8 9 3 1 0

(c) 2 5 6 7 4 8 9 3 1 0

0

(c) 2 5 6 7 4 8 9 3 1 0

0 1

(c) 2 5 6 7 4 8 9 3 1 0

0 1 2

(c) 2 5 6 7 4 8 9 3 1 0

0 1 ~~2~~

2

(c) 2 5 6 7 4 8 9 3 1 0

0 1 ~~2~~ 3

2

(c) 2 5 6 7 4 8 9 3 1 0

0 1 ~~2~~ 3 4

2

(c) 2 5 6 7 4 8 9 3 1 0

0 1 ~~2~~ 3 4 5

2

(c) 2 5 6 7 4 8 9 3 1 0

0 1 ~~2~~ 3 4 ~~5~~

2 5

(c) 2 5 6 7 4 8 9 3 1 0

0 1 ~~2~~ 3 4 ~~5~~ 6

2 5

(c) 2 5 6 7 4 8 9 3 1 0

0 1 ~~2~~ 3 4 ~~5~~ ~~6~~

2 5 6

(c) 2 5 6 7 4 8 9 3 1 0

0 1 ~~2~~ 3 4 ~~5~~ ~~6~~ 7

2 5 6

(c) 2 5 6 7 4 8 9 3 1 0

0 1 ~~2~~ 3 4 ~~5~~ ~~6~~ ~~7~~

2 5 6 7

(c) 2 5 6 7 4 8 9 3 1 0

0 1 ~~2~~ 3 ~~4~~ ~~5~~ ~~6~~ ~~7~~

2 5 6 7 4

(c) 2 5 6 7 4 8 9 3 1 0

0 1 ~~2~~ 3 ~~4~~ ~~5~~ ~~6~~ ~~7~~ 8

2 5 6 7 4

(c) 2 5 6 7 4 8 9 3 1 0

0 1 ~~2~~ 3 ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~

2 5 6 7 4 8

(c) 2 5 6 7 4 8 9 3 1 0

0 1 ~~2~~ 3 ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ 9

2 5 6 7 4 8

(c) 2 5 6 7 4 8 9 3 1 0

0 1 ~~2~~ 3 ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ ~~9~~

2 5 6 7 4 8 9

(c) 2 5 6 7 4 8 9 3 1 0

0 1 ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ ~~9~~

2 5 6 7 4 8 9 3

(c) 2 5 6 7 4 8 9 3 1 0

0 ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ ~~9~~

2 5 6 7 4 8 9 3 1

(c) 2 5 6 7 4 8 9 3 1 0

~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ ~~9~~

2 5 6 7 4 8 9 3 1 0

(c) 2 5 6 7 4 8 9 3 1 0

~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ ~~9~~

2 5 6 7 4 8 9 3 1 0 True

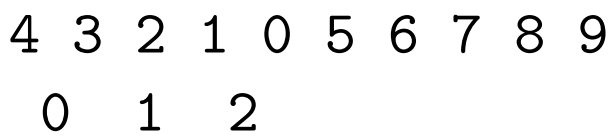
(d) 4 3 2 1 0 5 6 7 8 9

(d) 4 3 2 1 0 5 6 7 8 9

0

(d) 4 3 2 1 0 5 6 7 8 9

0 1

(d)


(d)

4	3	2	1	0	5	6	7	8	9
	0		1		2		3		

(d)

4	3	2	1	0	5	6	7	8	9
	0		1		2		3		4

(d) 4 3 2 1 0 5 6 7 8 9

0 1 2 3 ~~4~~

4

(d) 4 3 2 1 0 5 6 7 8 9

0 1 2 ~~3~~ ~~4~~

4 3

(d) 4 3 2 1 0 5 6 7 8 9

0 1 ~~2~~ ~~3~~ ~~4~~

4 3 2

(d) 4 3 2 1 0 5 6 7 8 9

0 ~~1~~ ~~2~~ ~~3~~ ~~4~~

4 3 2 1

(d) 4 3 2 1 0 5 6 7 8 9

~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~

4 3 2 1 0

(d) 4 3 2 1 0 5 6 7 8 9

~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5

4 3 2 1 0

(d) 4 3 2 1 0 5 6 7 8 9

~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~

4 3 2 1 0 5

(d) 4 3 2 1 0 5 6 7 8 9

~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ 6

4 3 2 1 0 5

(d) 4 3 2 1 0 5 6 7 8 9

~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~

4 3 2 1 0 5 6

(d) 4 3 2 1 0 5 6 7 8 9

~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ 7

4 3 2 1 0 5 6

(d) 4 3 2 1 0 5 6 7 8 9

~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~

4 3 2 1 0 5 6 7

(d) 4 3 2 1 0 5 6 7 8 9

~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ 8

4 3 2 1 0 5 6 7

(d) 4 3 2 1 0 5 6 7 8 9

~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~

4 3 2 1 0 5 6 7 8

(d) 4 3 2 1 0 5 6 7 8 9

~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ 9

4 3 2 1 0 5 6 7 8

(d) 4 3 2 1 0 5 6 7 8 9

~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ ~~9~~

4 3 2 1 0 5 6 7 8 9

(d) 4 3 2 1 0 5 6 7 8 9

~~0 1 2 3 4 5 6 7 8 9~~

4 3 2 1 0 5 6 7 8 9 True

Aufgabe 6

Angenommen ein Programm führt eine abwechselnde Folge von `enqueue`- und `dequeue`-Operationen auf einer Queue aus. Die `enqueue`-Operationen stellen dabei die ganzen Zahlen von 0 bis 9 in dieser Reihenfolge in die Warteschlange. Die `dequeue`-Operatioenn geben den Rückgabewert aus. Welche der folgenden Sequenzen kann es nicht geben?

(a) `0 1 2 3 4 5 6 7 8 9`

(b) `4 6 8 7 5 3 2 9 0 1`

(c) `2 5 6 7 4 8 9 3 1 0`

(d) `4 3 2 1 0 5 6 7 8 9`

Aufgabe 6

(a)	`0 1 2 3 4 5 6 7 8 9`	True
(b)	`4 6 8 7 5 3 2 9 0 1`	False
(c)	`2 5 6 7 4 8 9 3 1 0`	False
(d)	`4 3 2 1 0 5 6 7 8 9`	False

Wenn die Elemente in aufsteigender Reihenfolge in die Queue eingereiht werden, dann müssen sie auch in der gleichen Reihenfolge wieder die Queue verlassen (First In – First Out). Daher ist für die Inputfolge `[0, 1, 2, ..., 8, 9]` nur (a) möglich.

Aufgabe 7

(a) Welche Laufzeit $O(?)$ hat die im Unterricht besprochene Python-Implementierung von `enqueue(item)`?

(b) Welche Laufzeit $O(?)$ hat die im Unterricht besprochene Python-Implementierung von `dequeue()`?

(c) Mit welcher minimalen Laufzeit lässt sich bei der im Unterricht entwickelten Klasse ein Iterator implementieren, der alle Elemente der Queue durchläuft.

Aufgabe 7

(a) Welche Laufzeit $O(?)$ hat unsere Python-Implementierung von `enqueue(item)`?

Aufgabe 7

(a) Welche Laufzeit $O(?)$ hat unsere Python-Implementierung von `enqueue(item)`?

$O(1)$

Aufgabe 7

(a) Welche Laufzeit $O(?)$ hat unsere Python-Implementierung von `enqueue(item)`?

$O(1)$

(b) Welche Laufzeit $O(?)$ hat unsere Python-Implementierung von `dequeue()`?

Aufgabe 7

(a) Welche Laufzeit $O(?)$ hat unsere Python-Implementierung von `enqueue(item)`?

$O(1)$

(b) Welche Laufzeit $O(?)$ hat unsere Python-Implementierung von `dequeue()`?

$O(1)$

Aufgabe 7

(a) Welche Laufzeit $O(?)$ hat unsere Python-Implementierung von `enqueue(item)`?

$O(1)$

(b) Welche Laufzeit $O(?)$ hat unsere Python-Implementierung von `dequeue()`?

$O(1)$

(c) Mit welcher minimalen Laufzeit lässt sich ein Iterator implementieren, der alle Elemente der Queue durchläuft.

Aufgabe 7

(a) Welche Laufzeit $O(?)$ hat unsere Python-Implementierung von `enqueue(item)`?

$O(1)$

(b) Welche Laufzeit $O(?)$ hat unsere Python-Implementierung von `dequeue()`?

$O(1)$

(c) Mit welcher minimalen Laufzeit lässt sich ein Iterator implementieren, der alle Elemente der Queue durchläuft.

$O(n)$ wenn n die Länge der Queue bezeichnet.

Aufgabe 8

Vervollständige das Zeigerdiagram der Queue `q` anhand des Speicherabbildes, das noch andere Daten enthält. Ein Knotenobjekt besteht jeweils aus zwei benachbarten Feldern, wobei im ersten Feld der Wert und im zweiten die Referenz auf das erste Feld des nächsten Knoten steht. `00` steht für `None`. Welche Daten sind in der Queue?

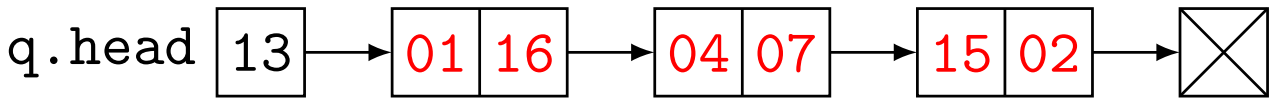
	0.	1.	2.	3.	4.	5.	6.	7.	8.	9.
0.		06	00	08	16	15	00	15	02	09
1.	15	11	18	01	16	06	04	07	16	00

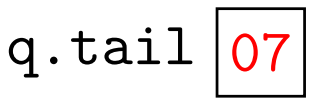
`q.head` [13]

`q.tail` [ ]

Aufgabe 8

	0.	1.	2.	3.	4.	5.	6.	7.	8.	9.
0.		06	00	08	16	15	00	15	02	09
1.	15	11	18	01	16	06	04	07	16	00





Aufgabe 9

Nenne zwei Anwendungen der Datenstruktur Queue.

Aufgabe 9

Anwendungen von Queues

- Als Puffer für Tastatur- und Mauseingaben
- Für die Interprozesskommunikation
- Druckerwarteschlangen
- Datenaustausch in Netzwerken (Daten, die an einen Router geschickt werden, weren üblicherweise in einer Queue zwischengespeichert, bevor sie weitergeleitet werden)

Aufgabe 10

Erkläre, wofür das Silbenkurzwort *Deque* genau steht.

Aufgabe 10

Deque steht für *Double Ended Queue*.

Aufgabe 11

Welche strukturellen Merkmale hat die Datenstruktur *Deque*?

Aufgabe 11

Es handelt sich um eine Sammlung von Elementen, in der

- die Reihenfolge der Elemente wesentlich ist,
- Elemente wiederholt vorkommen können,
- Die Elemente jeweils am Anfang *und* am Ende hinzugefügt und entnommen werden können.

Aufgabe 12

Gegeben ist ist ein Teil des Codes für eine Queue auf der Grundlage einer einfach verketteten Liste. Vervollständige den fehlenden Code des Konstruktors sowie der Methoden `__len__(...)` und `enqueue(...)`.

```
class Node:

    def __init__(self,obj,nxt=None):
        self.obj = obj
        self.nxt = None

class Queue:

    def __init__(self):
        self.head = None
        self.tail = None
        self.size = 0

    def __len__(self):
        return self.size
```

Aufgabe 12

```
class Node:

    def __init__(self,obj,nxt=None):
        self.obj = obj
        self.nxt = None

class Queue:

    def __init__(self):
        self.head = None
        self.tail = None
        self.size = 0

    def __len__(self):
        return self.size

    def enqueue(self,obj):
        new_node = Node(obj)
        if self.head == None:
            self.head = new_node
```