ALGORITHMS & DATASTRUCTURES

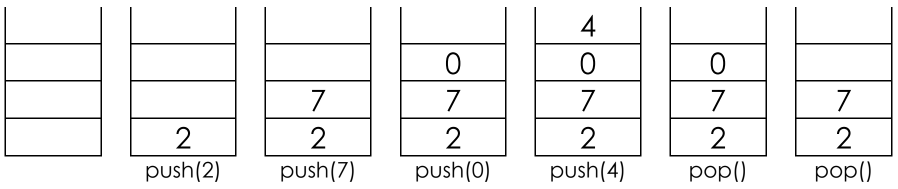
LES 2 : BASIS DATASTRUCTUREN

UITWERKING

THEORIE OPGAVEN

OPGAVE 1 (6.1AB)

a)



b)

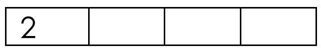
2	enqueue(2)
2-7	enqueue(7)
2-7-0	enqueue(0)
2-7-0-4	enqueue(4)
7-0-4	dequeue()
0-4	dequeue()

OPGAVE 2 (16.1)

a)

[2]
[1] -> [2]
[2]
[4] -> [2]
[3] -> [4] -> [2]
[4] -> [2]
[2]
[5] -> [2]

b)



2	1		

2			

2	4		

2	4	3	

2	4		

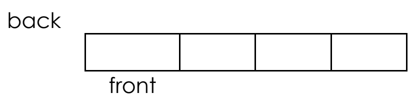
2			

2	5		

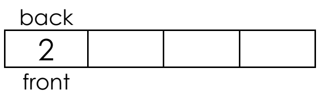
c)

[2]
[2] -> [1]
[1]
[1] -> [4]
[1] -> [4] -> [3]
[4] -> [3]
[3]
[3] -> [5]

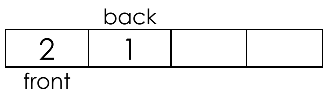
d)



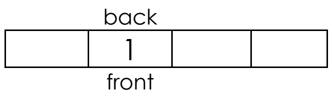
add(2)



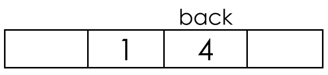
add(1)

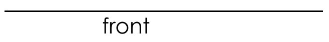


remove()

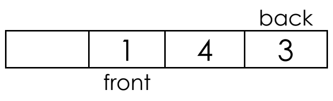


add(4)

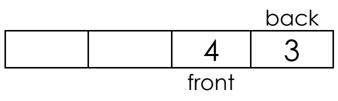




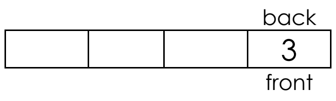
add(3)



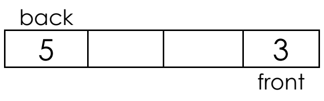
remove()



remove()

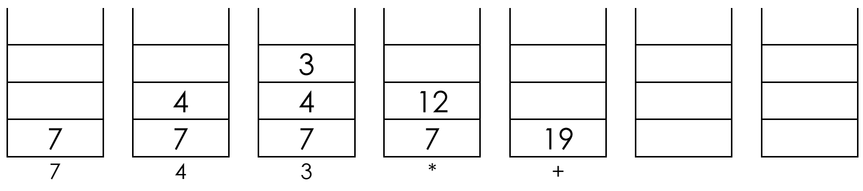


add(5)

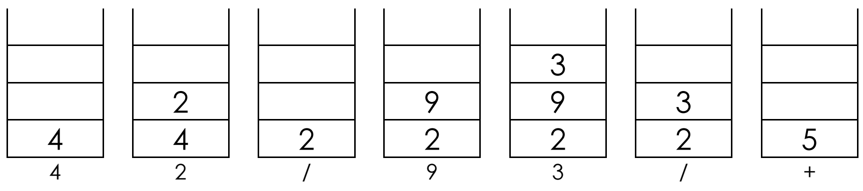


OPGAVE 3

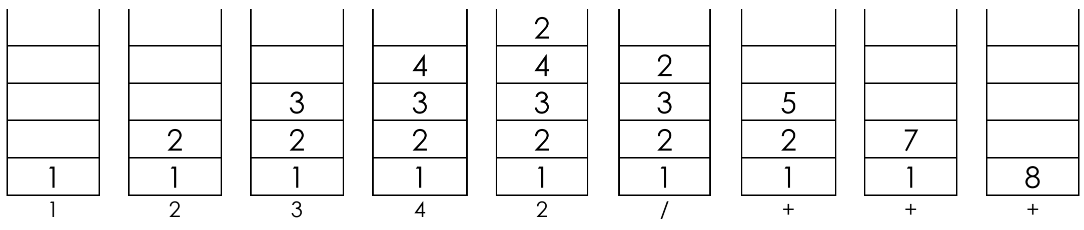
a) 743*+



b) 42/93/+



a) 12342/+++



PRAKTIJK

OPDRACHT 1 ARRAYLIST

a) -
b) Afhankelijk van implementatie!

void add(int n)	**O(c)**
int get(int index)	O(c)
void set(int index, int n)	O(c)
void print()	O(N)
void clear()	O(c)
int countOccurences(int n)	O(N)

OPDRACHT 2 LINKED LIST

a) -
b) Afhankelijk van implementatie!

void addFirst(T data)	**O(c)**
void clear()	O(c)
void print()	O(N)
void insert(int index, T data)	O(N)
void removeFirst()	O(c)
T getFirst()	O(c)

OPDRACHT 4 TOEPASSING STACK

Uitwerking staat ook in hoofdstuk 11!

OPDRACHT 5 QUEUE (16.6)

ArrayList is not a good choice to implement to a queue because when the array size is doubled the, ArrayList implementation copies elements at the same location. In the case of a queue (see array based implementation of queue), in the new doubled array, the elements are copied so that the front of the queue is at location 0. This allows the new element to be added at the end of the array. The ArrayList implementation will require extra work in this case.