Abstract Datatype

Papers Dock

COMPUTER SCIENCE 9618 PAPER 2

Abstract Datatype

An Abstract Datatype is a collection of data and set of operations on that data

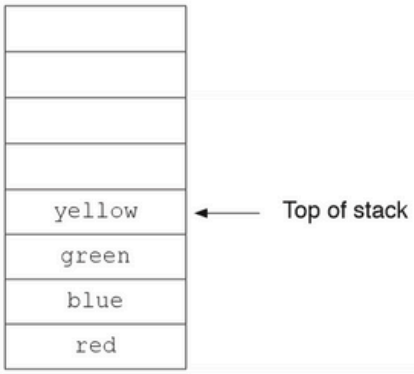
Stack Queue Linked List

Stacks

A list containing several items operating on the last in first out principle (LIFO).
Items can be added to the stack (PUSH) and removed from the stack (POP).
The first item added to the stack is the last item removed from the stack

Push("Taha")
Push("Ali")
Push("Amjad")
Push("Bano")
Pop()
Pop()
Push("Qasim")

1 **(a)** A stack contains the values `'red'`, `'blue'`, `'green'` and `'yellow'`.



(i) Show the contents of the stack in **part(a)** after the following operations.

```
POP()
PUSH('purple')
PUSH('orange')
```

[1]

(ii) Show the contents of the stack from **part(a)(i)** after these further operations.

```
POP()
POP()
PUSH('brown')
POP()
PUSH('black')
```

[1]

Question

Each Character from the word "Papersdock" is added into the stack one by one and when all the characters were popped out the string looked different. ("kcodsrepaP")

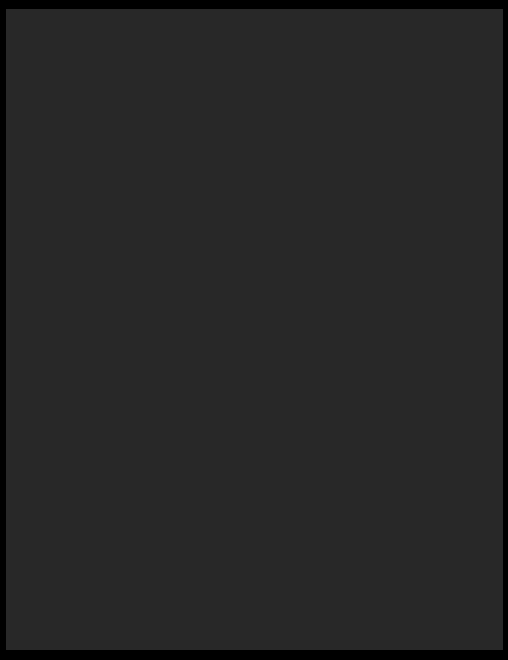
Answer

The Stack follows the principle of Last In First Out so the first item added in the stack is remved at the end so thats why the order is reversed

Queue

A list containing several items operating on the first in first out principle (FIFO).
The first item added is the first item remove from the queue
In queue the data is added from the rear end by using the EndPointer and removed from the front by using the StartPointer

Enqueue("Taha")
Enqueue("Ali")
Enqueue("Amjad")
Enqueue("Bano")
Dequeue()
Dequeue()
Enqueue("Qasim")



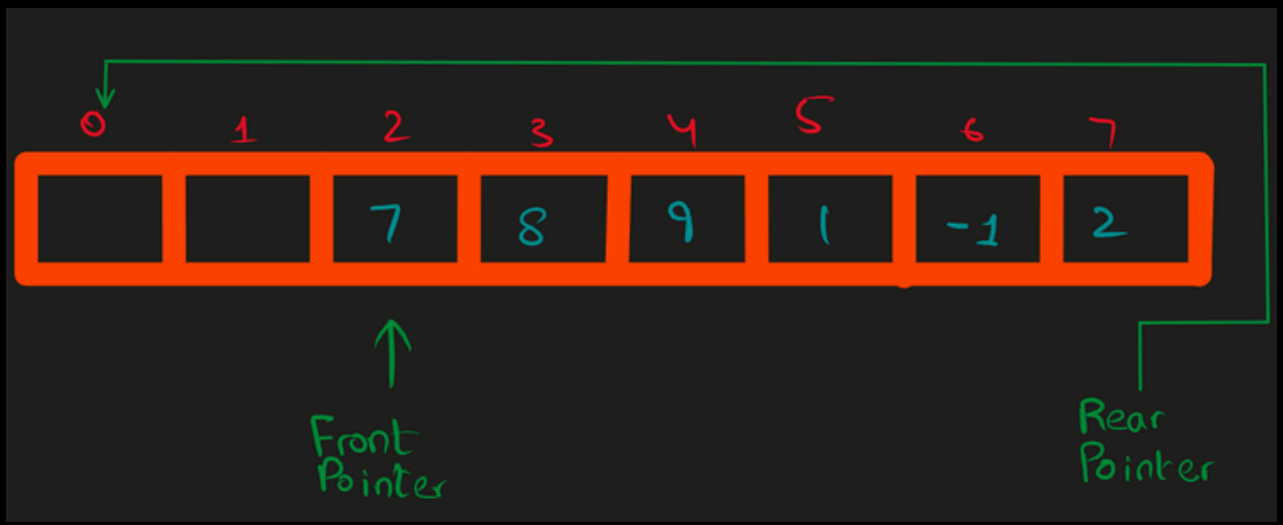
Linear Vs Circular

The condition for a linear queue being full is that rearpointer or the endpointer should point towards upperbound or the max index

Dequeue()
Dequeue()

According to the condition of queue being full is still true so if you want to enqueue a value it will still print "The Queue Is Full"

Circular Queue



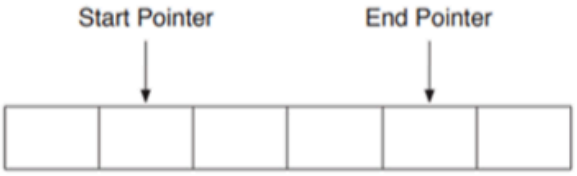
(b) When a student prints a document, a print job is created. The print job is sent to a print server.

The print server uses a queue to hold each print job waiting to be printed.

(i) The queue is circular and has six spaces to hold jobs.

The queue currently holds four jobs waiting to be printed. The jobs have arrived in the order A, B, D, C.

Complete the diagram to show the current contents of the queue.



[1]

(ii) Print jobs A and B are now complete. Four more print jobs have arrived in the order E, F, G, H.

Complete the diagram to show the current contents and pointers for the queue.

[3]

(iii) State what would happen if another print job is added to the queue in the status in **part (b)(ii)**.

.....

..... [1]

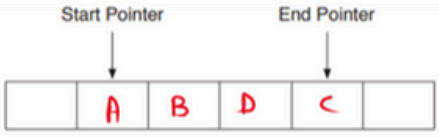
(b) When a student prints a document, a print job is created. The print job is sent to a print server.

The print server uses a queue to hold each print job waiting to be printed.

(i) The queue is circular and has six spaces to hold jobs.

The queue currently holds four jobs waiting to be printed. The jobs have arrived in the order A, B, D, C.

Complete the diagram to show the current contents of the queue.



[1]

(ii) Print jobs A and B are now complete. Four more print jobs have arrived in the order E, F, G, H.

Complete the diagram to show the current contents and pointers for the queue.

[3]

(iii) State what would happen if another print job is added to the queue in the status in **part (b)(ii)**.

An error Message.

..... [1]

3 **(a)** The diagram below represents a queue Abstract Data Type (ADT) that can hold a maximum of eight items.

The operation of this queue may be summarised as follows:

- The front of queue pointer points to the next item to be removed.
- The end of queue pointer points to the last item added.
- The queue is circular so that empty storage elements can be reused.

0	Frog	← Front of queue pointer
1	Cat	
2	Fish	
3	Elk	← End of queue pointer
4		
5		
6		
7		

(i) Describe how "Octopus" is added to the given queue.

.....

.....

.....

..... [2]

(ii) Describe how the next item in the given queue is removed and stored in the variable `AnimalName`.

.....

.....

.....

..... [2]

(iii) Describe the state of the queue when the **front of queue** and the **end of queue** pointers have the same value.

.....

..... [1]

(b) Some operations are carried out on the original queue given in **part (a)**.

(i) The current state of the queue is:

0	Frog
1	Cat
2	Fish
3	Elk
4	
5	
6	
7	

Complete the diagram to show the state of the queue after the following operations:

Add "Wasp", "Bee" and "Mouse", and then remove two data items.

[3]

(ii) The state of the queue after other operations are carried out is shown:

0	Frog	
1	Cat	
2	Fish	
3	Elk	← Front of queue pointer
4	Wasp	
5	Bee	
6	Mouse	← End of queue pointer
7	Ant	

Complete the following diagram to show the state of the queue after the following operations:

Remove one item, and then add "Dolphin" and "Shark".

0	
1	
2	
3	
4	
5	
6	
7	

[2]

(c) The queue is implemented using a 1D array.

Describe the algorithm that should be used to modify the **end of queue pointer** when adding an item to the queue.

Your algorithm should detect any potential error conditions.

.....

.....

.....

.....

.....

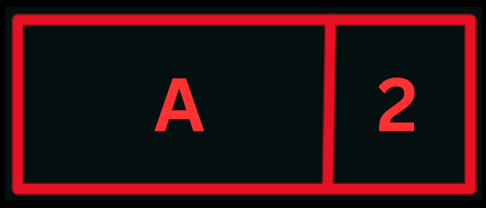
..... [3]

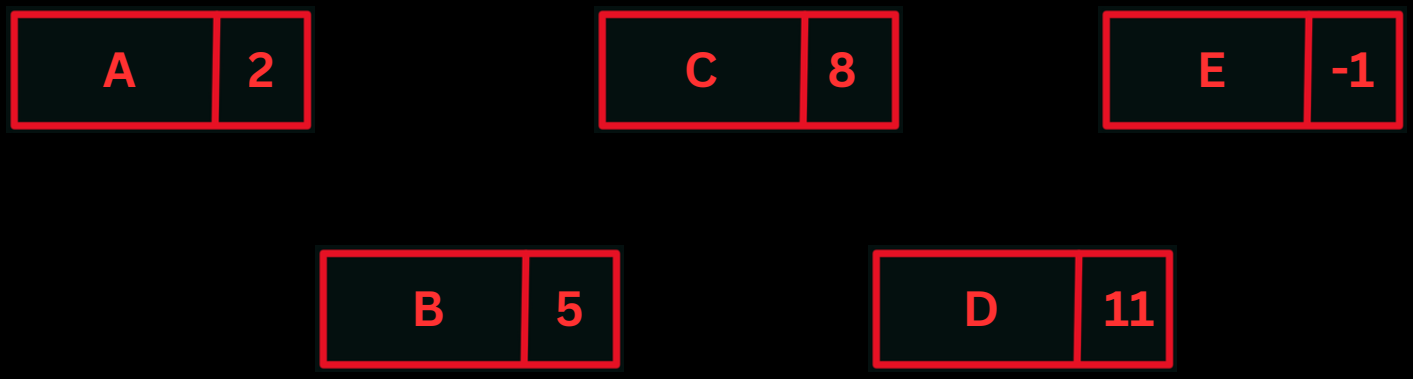
3(a)(i)	One mark per point: • `EoQ` pointer will move to point to location 4 // incremented `EoQ` (by 1) • Data value "Octopus" will be stored in location pointed to be `EoQ` / location 4	**2**
3(a)(ii)	One mark for each bullet • Value "Frog" // value pointed to by `FoQ` / location 0 is assigned to variable `AnimalName` • `FoQ` pointer will move to point to location 1 / point to "Cat" // incremented `FoQ` (by 1) **0** Frog ← **Front of queue pointer** **1** Cat **2** Fish **3** Elk ← **End of queue pointer**	**2**
3(a)(iii)	There is only one data item in the queue	**1**

3(b)(i)	One mark for data values plus one mark for pointers **0** Frog **1** Cat **2** Fish ← **Front of queue pointer** **3** Elk **4** **Wasp** **5** **Bee** **6** **Mouse** ← **End of queue pointer** **7** One mark for each pointer One mark for three new data values	**3**

3(b)(ii)	<table><tr><td>**0**</td><td>**Shark**</td><td>← **End of queue pointer**</td></tr><tr><td>**1**</td><td>(Cat)</td><td></td></tr><tr><td>**2**</td><td>(Fish)</td><td></td></tr><tr><td>**3**</td><td>(Elk)</td><td></td></tr><tr><td>**4**</td><td>Wasp</td><td>← **Front of queue pointer**</td></tr><tr><td>**5**</td><td>Bee</td><td></td></tr><tr><td>**6**</td><td>Mouse</td><td></td></tr><tr><td>**7**</td><td>**Dolphin**</td><td></td></tr></table> One mark for BOTH pointers One mark for all data values as shown	**2**
3(c)	One mark per point: 1 If incremented `EoQ` = `FoQ` then error condition: queue is full 2 Increment the `EoQ` 3 Manage wrap-around	**3**

Linked List





Ordered Linked List And Unordered Linked List

A linked list is a data structure used to store a collection of items where each item is linked to the next one using pointers.

There are two types of linked lists: ordered and unordered.

An ordered linked list is a list where the elements are arranged in ascending or descending.

an unordered linked list is a list where the elements are not sorted in any particular order.

1 An unordered linked list uses a 1D array to store the data.

Each item in the linked list is of a record type, `node`, with a field `data` and a field `nextNode`.

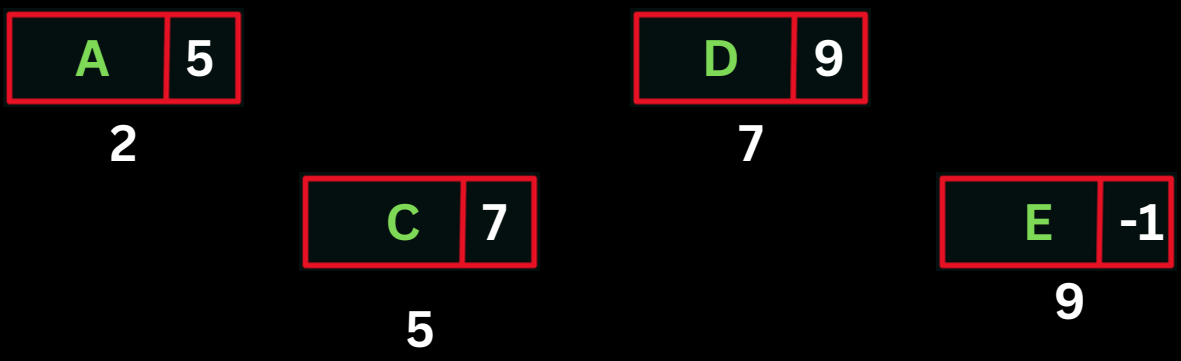
The current contents of the linked list are:

startPointer	0
emptyList	5

Index	data	nextNode
0	1	1
1	5	4
2	6	7
3	7	-1
4	2	2
5	0	6
6	0	8
7	56	3
8	0	9
9	0	-1

Linked List Insertion

S.P = 2

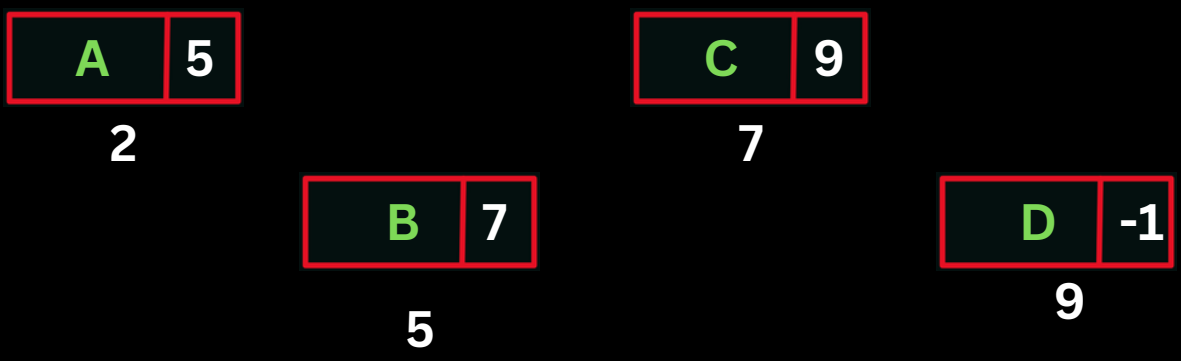


How To Insert A Node In A Linked List

- Check for a free node in a linked list
- Search for correct position
- Assign the Value B to the first node in free list
- Pointer from B will be changed to point towards C
- Pointer from A will point towards B
- Start Pointer in free list will move to point to next free node

Linked List Deletion

S.P = 2



How To Delete A Node In A Linked List

- Search for the node that you want to delete by incrementing the pointer and start from the first node.
- If the node that you want to delete is the first node, then point the start pointer to the next node in list.
- If the node that you want to delete is in the middle, then you would change the pointer value of the previous node and point it to the next node and point the free list pointer towards the node removed.