|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name: |  | Date |  | Class |  |

## Queues

|  |
| --- |
| **Word bank** |
| sequence | disk | optimum |
| keyboard | simulation | supermarket |
| First In First Out | size | printed |

A queue is a ....................... (FIFO) data structure. New elements may only be added to the end of a queue, and elements may only be retrieved from the front of a queue. The ....................... of data items is a queue is determined, therefore, by the order in which they are inserted. The ....................... of the queue depends on the number of items in it, just like a queue at traffic lights or at a ........................

Queues are used in a variety of applications:

* Output waiting to be ....................... is commonly stored in a queue on disk. In a room full of networked computers, several people may send work to be printed at more or less the same time. By putting the output into a queue on ......................., the output is printed on a first come, first served basis as soon as the printer is free.
* Characters typed at a keyboard are held in a queue in a ....................... buffer.
* Queues are useful in simulation problems. A ....................... program is one which attempts to model a real-life situation so as to learn something about it. An example is a program that simulates customers arriving at random times at the check-outs in a supermarket store, and taking random times to pass through the checkout. With the aid of a simulation program, the ....................... number of check-out counters can be established.

## Operations on a queue

|  |
| --- |
| **Word bank** |
| items | removed | Jason |
| queue | collection | joins |

The abstract data type queue is defined by its structure and the operations which can be performed on it. It is described as an ordered ....................... of ....................... which are added at the rear of the ......................., and ....................... from the front.



When Eli leaves the queue, the front pointer is made to point to .......................; the elements themselves do not move. When Adam ....................... the queue, the rear pointer points to Adam. Think of a queue in a doctor’s surgery – people leave and join the queue, but no one moves chairs.



The following queue operations are needed and an example is in the table below:

* enQueue(item) Add a new item to the rear of the queue
* deQueue() Remove the front item from the queue and return it
* isEmpty() Test to see whether the queue is empty
* isFull() Test to see whether queue is full

|  |  |  |
| --- | --- | --- |
| **Queue operation** | **Queue contents** | **Return value** |
| q.isEmpty | [] | True |
| q.enQueue(“Blue”) | [“Blue”] | (none) |
| q.enQueue(“Red”) | [“Blue”, “Red”] | - |
| q.enQueue(“Green”) | [“Blue”, “Red”, “Green”] | - |
| q.isFull() | [“Blue”, “Red”, “Green”] | False |
| q.isEmpty() | [“Blue”, “Red”, “Green”] | False |
| q.deQueue() | [“Red”, “Green”] | “Blue” |
| q.enQueue(“Yellow”) | [“Red”, “Green”, “Yellow”] | - |

## Question

Complete the **grey areas** in the following table to show the queue contents and the value returned by the function of method. The queue is named q.

|  |  |  |
| --- | --- | --- |
| **Queue operation** | **Queue contents** | **Return value** |
|  | [] | True |
|  | [“COD”] | - |
| q.enQueue(“Battlefield”) |  | - |
| q.isEmpty | [“COD”, “Battlefield”] |  |
| q.enQueue(“Halo”) | [“COD”, “Battlefield”, “Halo”] | - |
|  | [“Battlefield”, “Halo”] | “COD” |
| q.isEmpty() | [“Battlefield”, “Halo”] | False |
| q.deQueue() |  |  |
|  | [“Halo”, “Left 4 Dead 2”] | - |

## Question

Based on the operations above, create your own queue with your own data

|  |  |  |
| --- | --- | --- |
| **Queue operation** | **Queue contents** | **Return value** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Implementing a linear queue

|  |
| --- |
| **Word bank** |
| processing | rear | integer |
| one | filled | number |

There are basically two ways to implement a linear queue in an array or list:

1. As items leave the queue, all of the other items move up ....................... space so that the front of the queue is always the first element of the structure, e.g. q[0]. With a long queue, this may require significant ....................... time.
2. A linear queue can be implemented with pointers to the front and ....................... of the queue. An ....................... holding the size of the array (the maximum size of the queue) is needed, as well as a variable giving the ....................... of items currently in the queue. However, clearly a problem will arise as many items are added to and deleted from the queue, as space is created at the front of the queue which cannot be ......................., and items are added until the rear pointer points to the last element of the data structure.

## Question

1. The queue of names pictured below containing Jason, Milly, Bob and Adam has space for six names. What will be the situation when Jason and Milly leave the queue and jack joins it?

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 |
|  | Jason | Milly | Bob | Adam |  |

front = 1

rear = 4

1. How many names are now in the queue?

1. How many free spaces are left?

## A circular queue

|  |
| --- |
| **Word bank** |
| programmer | rather | being |
| dynamic | element | need |

One way of overcoming the limitations of implementing a queue as a linear queue is to use a circular queue instead, so that when the array fills up and the rear pointer points to the last ....................... of the array, say q[5], it will be made to point to the first element, q[0], when the next person joins the queue, assuming this element is empty. This solution requires some extra effort on the part of the ......................., and is less flexible than a ....................... data structure if the maximum number of items is not known in advance.



It is the pointers that move ....................... than the data, however with a circular queue the first items of data can be seen as ....................... next of the last item of data. A common implementation is for buffering, when items of data ....................... to be stored temporarily while they are waited to be passed to/from a device.

## Example:

The circular queue makes use of the spaces that are freed up at the front of a queue after they have been removed. It does this by wrapping the rear pointer around the array starting at the element 0 once the queue becomes full. If we start with this queue, the front pointer is 0 and the rear pointer is 4:

**FP** = Front pointer

**RP** = Rear pointer

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **FP** |  |  |  | **RP** |  |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Arsenal | Barcelona | City | Dortmund | Leicester |  |  |  |  |

If two items are removed, the queue will look like this:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **FP** |  | **RP** |  |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|  |  | City | Dortmund | Leicester |  |  |  |  |

Four new items are now added to the queue: “Liverpool”, “Madrid”,“Everton”and “United”. Notice that the rear pointer is now on 8

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **FP** |  |  |  |  |  | **RP** |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|  |  | City | Dortmund | Leicester | Liverpool | Madrid | Everton | United |

As this is a circular queue, the rear pointer can now wrap back around to the beginning. If a further item is added, the rear pointer would move to the position 0 as this is free to add “Stoke”.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **RP** |  | **FP** |  |  |  |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Stoke |  | City | Dortmund | Leicester | Liverpool | Madrid | Everton | United |

## Question

A circular queue is implemented in a fixed size array of six elements, indexed from 0. Show the contents of the queue and the **front** and **rear pointers for** a circular queue of 6 items when:

1. It is empty
2. Ali, Ben, Charlie, Davina, Enid, Fred join the queue. Ali, Ben and Charlie leave and Greg joins the queue.

## Priority queues

|  |
| --- |
| **Word bank** |
| determined | shorter | back |
| joins | inserted | operating |

In some situations where items placed in a queue, a system of priorities is used. For example an ......................., system might schedule jobs in order of priority, or a printer may give ......................., print jobs priority over longer ones.

A priority queue acts like a queue on that items are dequeued by removing them from the front of the queue. However, the logical order of the items within the queue is ......................., by their priority, with the highest priority items at the front of the queue and the lowest priority at the .......................,. It is therefore possible that a new item ......................., the queue at the front, rather than the rear.

Such a queue could be implemented by checking the priority of each item in the queue, starting at the rear and moving it along one place until an item with the same or lower priority is found, at which point the new item can be .......................,.

## Questions

1. What is meant by a **dynamic data structure** and why an inbuilt dynamic data structure in a programming language may be useful in implementing a queue (include an explanation of what is meant by **heap** in this context.
2. Print jobs are put in a queue to be printed. The queue is implemented in an array, indexed from 0, as a circular queue which can hold 5 jobs. Jobs enter the queue in the sequence Job1, Job2,Job3, Job4, Job5. Pointers **front** and **rear** point to the first and last items in the queue respectively.
3. Draw a diagram to show how the print jobs are stored. Include pointers in your diagram.
4. Two jobs are printed and leave the queue. Another job, job6 joins the queue. Draw a diagram representing the new situation