Date 07/06/2023

Lab Sheet 04

① What is circular queue ?

A circular queue is the extended version of a regular queue where the last element is connected to the first element.

② What are the characteristics of circular queue ?

* A Circular queue is the extended version of regular queue where the last element is connected to the first element.

* Thus forming a circle - like structure.

* The circular queue solves the major limitation of the normal queue.

* In a normal queue, after a bit of insertion and deletion, There will be non usable - empty space

③ Give applications of circular queue

01. Memory Management :
circular queue is used in memory management

02. Process Sheduling :
A CPU uses a queue to schedule processes

03. Trafic Systems :
Quesies are ~~so in~~ also used in trafic elements.

SANVIN

④ What is the algorithm of circular queue?

A circular queue is a type of data structure that implements a queue using a fixed size array. By enabling items to be inserted and removed from both ends of the queue without shifting elements, it gets arround the drawbacks of a conventional linear queue. Initializing the queue, adding components, removing elements and testing for different circumstances like a full or empty queue are all part of the circular algorithm.

⑤ Write a simple program of circular

```
class CircularQueue:
    def __init__(self, size):
        self.size = size
        self.front = -1
        self.rear = -1
        self.queue = []

    def __enqueue(self, value):
        if self.is_full():
            print ("Queue is full")
            return
        if self.front = -1:
            self.front = 0
        else:
            self.rear = (self.rear + 1) % self.size
        self.queue.append(value)

    def __dequeue(self):
```

SANVIN

```
        if self.is_empty():
            print("Queue is empty")
            return

        value = self.queue[self.front]
        self.queue.pop(self.front)

        if self.front == self.rear:
            self.front = -1
            self.rear = -1
        return value

    def is_full(self):
        return self.rear == (self.front
                -1 + self.size) % self.size

    def is_empty(self):
        return self.front == -1

    def __str__(self):
        return str(self.queue)
```

```
if __name__ = "__main__":
        queue = CircularQueue(5)
        queue.enqueue(1)
        queue.enqueue(2)
        queue.enqueue(3)
        print(queue)
        queue.dequeue()
        print(queue)
        queue.enqueue(4)
        print(queue)
```

⑥ Compare and contrast linear queues and circular queues.

Feature	Linear Queue	Circular Queue
Data Structure	Linear	Circular
Insertion	Rear end	Either end
Deletion	Front	Either end
Overflow	Possible	Not possible
Underflow	Possible	Not possible
Memory efficiency	Less efficient	More efficient
Applications	* CPU scheduling * fire buffering	* Memory management * Array rotation

Here are some points of comparison :-

* Complexity
  Circular queues are more complex to implement than linear queues, because they need to keep track of the wraparound point
* Performance
  Circular queues are generally more performant than linear queues, because they do not have to worry about overflow.
* Memory Usage
  Circular queues use less memory than linear queues, because they do not need to store empty slots.