**Aim:** Implement Linear Queue ADT using an array.

#include <limits.h>

#include <stdio.h> #include <stdlib.h>

struct Queue {

int front, rear, size; unsigned capacity; int\* array;

};

struct Queue\* createQueue(unsigned capacity)

{

struct Queue\* queue = (struct Queue\*)malloc( sizeof(struct Queue));

queue->capacity = capacity; queue->front = queue->size = 0;

queue->rear = capacity- 1; queue->array = (int\*)malloc( queue->capacity \* sizeof(int)); return queue;

}

int isFull(struct Queue\* queue)

{

return (queue->size == queue->capacity);

}

int isEmpty(struct Queue\* queue)

{

return (queue->size == 0);

}

void enqueue(struct Queue\* queue, int item)

{

if (isFull(queue)) return;

queue->rear = (queue->rear + 1)

% queue->capacity;

queue->array[queue->rear] = item; queue->size = queue->size + 1;

printf("%d enqueued to queue\n", item);

}

int dequeue(struct Queue\* queue)

{

if (isEmpty(queue)) return INT\_MIN;

int item = queue->array[queue->front]; queue->front = (queue->front + 1)

% queue->capacity;

queue->size = queue->size- 1; return item;

}

int front(struct Queue\* queue)

{

if (isEmpty(queue)) return INT\_MIN;

return queue->array[queue->front];

}

int rear(struct Queue\* queue)

{

if (isEmpty(queue)) return INT\_MIN;

return queue->array[queue->rear];

}

int main()

{

struct Queue\* queue = createQueue(1000); enqueue(queue, 10);

enqueue(queue, 20);

enqueue(queue, 30);

enqueue(queue, 40);

printf("%d dequeued from queue\n\n", dequeue(queue));

printf("Front item is %d\n", front(queue)); printf("Rear item is %d\n", rear(queue)); return 0;

}

**Conclusion:** This experiment delves into implementing a Linear Queue ADT using an array. It provides insights into how queues work, including enqueue, dequeue, and front operations, and the utilization of arrays for first-in-first-out (FIFO) data structures.