```
// CPP program for array
// implementation of queue
#include <bits/stdc++.h>

using namespace std;

// A structure to represent a queue

class Queue {

public:

    int front, rear, size;

    unsigned capacity;

    int* array;
};

// function to create a queue
// of given capacity.
// It initializes size of queue as 0
Queue* createQueue(unsigned capacity)
{

    Queue* queue = new Queue();

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



    // This is important, see the enqueue

    queue->rear = capacity - 1;

    queue->array = new int[queue->capacity];

    return queue;
}

// Queue is full when size
// becomes equal to the capacity

int isFull(Queue* queue)
{

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

// Queue is empty when size is 0

int isEmpty(Queue* queue)
{

    return (queue->size == 0);
}
```

```
// Function to add an item to the queue.

// It changes rear and size


void enqueue(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;


    cout << item << " enqueued to queue\n";

}


// Function to remove an item from queue.

// It changes front and size


int dequeue(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;
}

// Function to get front of queue

int front(Queue* queue)
{

    if (isEmpty(queue))

        return INT_MIN;

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

// Function to get rear of queue

int rear(Queue* queue)
{
    if (isEmpty(queue))

        return INT_MIN;

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

// Driver code

int main()
{

    Queue* queue = createQueue(1000);



    enqueue(queue, 10);

    enqueue(queue, 20);

    enqueue(queue, 30);

    enqueue(queue, 40);



    cout << dequeue(queue)

         << " dequeued from queue\n";
    cout << "Front item is "

         << front(queue) << endl;

    cout << "Rear item is "

         << rear(queue) << endl;



    return 0;
}
```