```
// Justin Dang Student ID: 1148267
/*
creates a LinkedQueue (int)array
*/
#include <iostream>
using namespace std;
#define MaxSize 10


class arrayQueue {
private:
    int front,  // holds node number of the front node(0)
        back,   // holds node number of the back node(x) x = any number between 0-10
        nodeCount;      // holds the amount of nodes created
    int queue[MaxSize]; // array where the ints will be stored
public:
    arrayQueue() { front = back = nodeCount = 0; } // queue contstructor
    bool isEmpty() { return nodeCount == 0; }      // returns true if there are no nodes
    bool enqueue(int data) {
        if (nodeCount++ == MaxSize) { // catches error if the node count is 1 less than
            nodeCount--;              // MaxSize of our array
            cout << "Maximum queue size hit.\n\n";
            cout << "Front: " << front << endl
                << "Back: " << back << endl
                << "nodeCount: " << nodeCount << "\n\n";
            return false;
        }

        queue[back++] = data; // queues the integer at the back of the array

        if (back >= MaxSize)  // used to wrap back to the front of the array
            back = 0;
        return true;
    }
    int dequeue() {
        if (isEmpty()) { // catches error if the amount of nodes we have are 0
            cout << "Queue is empty.\n\n";
            return -999;
        }

        nodeCount--;                    // removes 1 from node count to reflect action
        int intReturn = queue[front++]; // stores the front int into a temp

        if (front > MaxSize)            // rotates the queue to the front of array
            front = 0;

        return intReturn;
    }
    void print() {
        cout << "From front to back, the queue values are: ";
        for (int i = front; i < back; i++)
            cout << queue[i] << ' ';
        cout << "\n\n";
    }
};
int main()
{
    cout << "Creating a linkedQueue array. \n\n\n";
    arrayQueue *array1;
    array1 = new arrayQueue();
    cout << "Enqueue 15.\n\n";
    array1->enqueue(15);
    array1->print();

    cout << "Enqueue 3.\n\n";
    array1->enqueue(3);
    array1->print();

    cout << "Enqueue 5.\n\n";
    array1->enqueue(5);
    array1->print();

    cout << "Enqueue 10.\n\n";
    array1->enqueue(10);
    array1->print();

    cout << "----------------------------------\nEmptying Queue. . .\n\n";
    // removes every node in the array until an error is thrown
    do{
        cout << "Removed " << array1->dequeue() << " from the array." << "\n\n";
        array1->print();
    } while (!array1->isEmpty());
    array1->dequeue();

    // fills every node in the array until error is thrown
    cout << "----------------------------------Filling queue. . .\n\n";
    while (array1->enqueue(5))
        cout << '.';
    array1->print();
}
/*//----------------------------------case 1:
Creating a linkedQueue array.


Enqueue 15.

From front to back, the queue values are: 15

Enqueue 3.
```

```

From front to back, the queue values are: 15 3

Enqueue 5.

From front to back, the queue values are: 15 3 5

Enqueue 10.

From front to back, the queue values are: 15 3 5 10

----------------------------------
Emptying Queue. . .

Removed 15 from the array.

From front to back, the queue values are: 3 5 10

Removed 3 from the array.

From front to back, the queue values are: 5 10

Removed 5 from the array.

From front to back, the queue values are: 10

Removed 10 from the array.

From front to back, the queue values are:

Queue is empty.

----------------------------------Filling queue. . .

..........Maximum queue size hit.

Front: 4
Back: 4
nodeCount: 10

From front to back, the queue values are:


*///----------------------------------


```