```
/*****************************************************************************
*
*  Author:           Miklos Moreno
*  Email:            miklosam1999@gmail.com
*  Label:            A04
*  Title:            Basic Project Organization
*  Course:           CMPS 2143
*  Semester:         Fall 2021
*
*  Description:
*        This is a program created by prof Griffin. We are to comment it in
*        a style acceptable by the guidelines given.
*
*  Usage:
*        none right now
*
*  Files:            main.cpp
*****************************************************************************/
#include <iostream>

using namespace std;

/**
 * Circular Array Que
 *
 * Description:
 *      a que that keeps track of the rear.
 *
 * Public Methods:
 *      - constructor   CircularArrayQue()
 *      - constructor   CircularArrayQue(int size)
 *      - void          Push(int item)
 *      - int           Pop()
 *
 * Private Methods:
 *      - int            *Container
 *      - int            Front
 *      - int            Rear
 *      - int            QueSize
 *      - int            CurrentSize
 *      - void           init(int size = 0)
 *      - bool           Full()
 *
 *
 * Usage:
 *
 *      - CircularArrayQue()            // create queue of size 10
 *      - CircularArrayQue(int size)    // create queue of size given by user
 *      - Push(int item)                // push int onto end of queue
 *      - Pop()                         // pop off int from front
 *      - init(int size = 0)            // enters values of 0 into queue
 *      - Full()                        // boolean checks if queue if full
```

```
 *
 */
class CircularArrayQue
{
private:
    int *Container;
    int Front;
    int Rear;
    int QueSize; // items in the queue
    int CurrentSize;
    void init(int size = 0)
    {
        Front = Rear = CurrentSize = 0;
        QueSize = size;
    }

    /**
     * Private : Full
     *
     * Description:
     *      returns bool value checking if the queue is full
     *
     * Params:
     *      none
     *
     * Returns:
     *      returns T of F
     */
    bool Full()
    {
        return CurrentSize == QueSize;
    }

public:
    /**
     * Public : CircularArrayQue
     *
     * Description:
     *      Constructs an empty queue.
     *
     * Params:
     *      none
     *
     * Returns:
     *      creates queue
     */
    CircularArrayQue()
    {
        Container = new int[10];
        init(10);
    }

    /**
     * Public : CircularArrayQue
```

```
     *
     * Description:
     *      Constructs an empty queue.
     *
     * Params:
     *      int : queue size
     *
     * Returns:
     *      creates queue
     */
    CircularArrayQue(int size)
    {
        Container = new int[size];
        init(size);
    }

    /**
     * Public : Push
     *
     * Description:
     *      pushed an interger onto the queue
     *
     * Params:
     *      int : item pushed onto the queue
     *
     * Returns:
     *      void
     */
    void Push(int item)
    {
        if (!Full())
        {
            Container[Rear] = item;
            Rear = (Rear + 1) % QueSize;
            CurrentSize++;
        }
        else
        {
            cout << "FULL!!!!" << endl;
        }
    }

    /**
     * Public : Pop
     *
     * Description:
     *      Removes item at front of the queue
     *
     * Params:
     *      none
     *
     * Returns:
     *      int : item from queue that was just removed
     */
```

```
    int Pop()
    {
        int temp = Container[Front];
        Front = (Front + 1) % QueSize;
        CurrentSize--;
        return temp;
    }
    friend ostream &operator<<(ostream &os, const CircularArrayQue &other);
};

/**
     * Public : Name? Operator maybe
     *
     * Description:
     *      Replaces print function and gives compiler method to cout the queue
     *
     * Params:
     *      ostream : outfile
     *      const   : queue (const so it cant be changed)
     *
     * Returns:
     *      ostream : outfile
     */
ostream &operator<<(ostream &os, const CircularArrayQue &other)
{

    for (int i = other.Front; i < other.CurrentSize; i = (i + 1) % other.QueSize)
    {
        os << other.Container[i] << " ";
    }
    os << endl;
    return os;
}

/**
 * Main Driver
 *
 * For this program, the main driver was used to test the CircularArrayQue class
 *
 */
int main()
{
    CircularArrayQue C1(5); // creates queue of size 5

    // C1.Push(34);
    // C1.Push(38);
    // C1.Push(44);
    // C1.Push(22);
    // C1.Push(99);
    // C1.Push(100);

    C1.Push(1); // push int 1 onto queue
    C1.Push(2); // push int 2 onto queue
    C1.Push(3); // push int 3 onto queue
```

```
    // C1.Push(6);
    // C1.Push(7);
    cout << C1 << endl; // print queue 'C1'

    // C1.Push(1);
    // C1.Push(2);
    // C1.Push(3);

    cout << C1 << endl; // print queue 'C1'
}
```