```cpp
//hesham omar 20200060
//capitalize and sort string using queues

#include <iostream>
using namespace std;

class Node
{
public:
	char data;
	Node* next;
	Node()
	{
		data = '\0';
		next = NULL;
	}
};

class Queue
{
public:
	Node* front;
	Node* rear;

	Queue()
	{
		front = rear = NULL;
	}

	bool isEmpty()
	{
		if (front == NULL)//&& rear == NULL)
			return true;
		else
			return false;

	}

	void Enqueue(char item)
	{
		Node* newnode = new Node();
		newnode->data = item;

		if (isEmpty())
			front = rear = newnode;
		else
		{
			rear->next = newnode;
			rear = newnode;
		}
	}

	void display()
	{
		if (isEmpty())
			cout << "Queue is Empty, no items to display \n";
		else
		{
			Node* temp = front;
			while (temp != NULL)
```

```
                {
                        cout << temp->data << " ";
                        temp = temp->next;
                }
                cout << endl;
        }
    }

    int count()
    {
        int counter = 0;
        Node* temp = front;
        while (temp != NULL)
        {
                counter++;
                temp = temp->next;
        }
        return counter;
    }

    int Dequeue()
    {
        int delvalue = -1;
        if (isEmpty())
                cout << "The queue is empty \n";
        else if (front == rear)
        {
                delete front;
                front = rear = NULL;
        }
        else
        {
                Node* delptr = front;
                front = front->next;
                delvalue = delptr->data;
                delete delptr;
        }

        return delvalue;
    }

    int getFront()
    {
        return front->data;
    }
};

void sortQueue(Queue q)
    {
        int n = q.count();

        for (int i = 0; i < n; i++) {

        int minIndex = -1;
        int minValue = INT_MAX;
        for (int j = 0; j < n; j++) {
                int currValue = q.getFront();
                q.Dequeue();

                if (currValue < minValue && j < (n - i)) {
```

```
                        minValue = currValue;
                        minIndex = j;
                }
                q.Enqueue(currValue);
        }

                for (int j = 0; j < n; j++) {
                        int currValue = q.getFront();
                        q.Dequeue();
                        if (j != minIndex) {
                                q.Enqueue(currValue);
                        }
                }
        q.Enqueue(minValue);
}
        for (int i = 0; i < n; i++) {
        int curr = q.getFront();
        q.Dequeue();
        cout << char(curr) << " ";
        q.Enqueue(curr);
}
        cout << endl;
}

int main()
{
        Queue q1;
        string s = "zhYauc@MapaOwp!";

        int i = 0;
        while (s[i]!='\0')
        {
                q1.Enqueue(s[i]);
                i++;
        }

        cout << "Original string:\n";
        q1.display();

        int size = q1.count();
        while (size--)
        {
                char c = q1.Dequeue();
                char n = toupper(c);
                q1.Enqueue(n);
        }

        cout << "\nCapitalized string:\n";
        q1.display();

        cout << "\nCapitalized and sorted string:\n";
        sortQueue(q1);
}
```