/**********

Name: Noah Buchanan

Username: ua203

Problem Set: PS7

Due Date: July 17, 2020

**********/

Data Structures

July 16, 2020

1. (a) Queue implementing arrays is slower at its worse case and must be initialized before being used, Linked Lists require no initialization and must never resize so all methods run on O(1) however you may not be able to access locations in the middle with looping therefore causing a different scenario for the algorithm to run in linear time.

   (b) It all runs in constant time O(1) except for a search function for a linked list and the insert method for the array because it must contain a resize condition that will run on O(n).

   (c) Different requirements of the specific scenario may make one a better option, for example if you know how large your queue is going to be and it will never go over that number I would most likely opt for an array implementation of the queue.

2. (a) A waitlist on a website or app, first come first serve is essentially the FIFO policy used in queues.

   (b) An operating system that queues processes in order by priority so most important is put in first to be executed and then executed first, or in other words CPU scheduling.

   (c) A pathfinding algorithm that weights the different paths by length and stores them in a queue based on that so that it keeps track of which paths it has not explored yet and which are the shortest to find a path in the minimal amount of time.

1 UAQueueLinkedList

I removed all comments on the classes to make it easier to read

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class UAQueueLinkedList<T> {

        int size;
        Wrapper<T> head;
        Wrapper<T> tail;

        public void enqueue(T s) {

                Wrapper<T> temp = new Wrapper<T>();
                temp.node = s;
                if (tail == null) {
                        tail = temp;
                        head = temp;
                } else {
                        tail.next = temp;
                        tail = temp;
                }
                size++;
        }

        public T dequeue() {

                T temp = head.node;
                head = head.next;
                size--;
                return temp;
        }

        public int size() {
                return size;
        }

        public boolean isQueueEmpty() {
                return size == 0;
        }

        public static class UAStudent<T> {
```

```
private int studentId;
private String firstName;
private String lastName;


public UAStudent(String input) {

        String[] x = input.split(",");

        this.lastName = x[0];
        this.firstName = x[1];
        this.studentId = Integer.parseInt(x[2]);
}

public UAStudent() {

}

public int getStudentId() {
        return studentId;
}

public void setStudentId(int studentId) {
        this.studentId = studentId;
}

public String getFirstName() {
        return firstName;
}

public void setFirstName(String firstName) {
        this.firstName = firstName;
}

public String getLastName() {
        return lastName;
}

public void setLastName(String lastName) {
        this.lastName = lastName;
}

public String toString() {
        return "Student: \t" + lastName + ", " + firstName;
}
```

```
}

public static void main(String[] args) throws IOException {

        UAQueueLinkedList<UAStudent> q = new UAQueueLinkedList<UAStudent>();

        BufferedReader br = new BufferedReader(new FileReader("records.txt"));

        String line = "";

        while ((line = br.readLine()) != null) {

                UAStudent x = new UAStudent(line);
                q.enqueue(x);
        }

        System.out.println("=== Start ==============================");
        System.out.println("Size of Queue:  " + q.size());

        System.out.println("\n\n");
        System.out.println("=== Inserts ============================");

        UAStudent a = new UAStudent("Mackey,Andrew,44444");
        q.enqueue(a);
        q.enqueue(new UAStudent("Mackey,Andrew,55555"));
        q.enqueue(new UAStudent("Mackey,Andrew,99999"));

        System.out.println("Size of Queue:  " + q.size());

        System.out.println("\n\n");
        System.out.println("=== Deletes ============================");
        System.out.println(q.dequeue());
        System.out.println(q.dequeue());

        System.out.println("Size of Queue:  " + q.size());

        System.out.println(q.dequeue());

        System.out.println("Size of Queue:  " + q.size());

        System.out.println("\n\n");
        System.out.println("=== Demonstrating Other Types ==========");
        UAQueueLinkedList<String> q2 = new UAQueueLinkedList<String>();
        q2.enqueue("test1");

        line = "";
```

```
                while ((line = br.readLine()) != null) {

                        q2.enqueue(line);
                }

                q2.enqueue("test2");

                while (!q2.isQueueEmpty()) {
                        System.out.println(q2.dequeue());
                }

                System.out.println("End");

        }

}
```

2 Wrapper

```
public class Wrapper<T> {

        T node;
        Wrapper<T> next;

}
```