```
/*
 * Author Name: Hunter Green
 * Email: hungree@okstate.edu
 * Date: 10/19/2023
 * Program Description: Data Structures Assignment3. Implements a .html syntax checker using a stack that is
 *                                                  implemeented using two queues
 */

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.util.ArrayList;



public class Main {
    public static void main(String[] args)throws Exception{

        String filePath = System.getProperty("user.dir") + "/" + args[0];
        ArrayList<String> tagArray = new ArrayList<>();
        Stack<String> stack = new Stack<>();
        char[] textAsCharArray = fileToArray(filePath); // stores the file as a char array

        // stores the file as a string array where each element is a line from the file
        String[] textAsStringArray = fileToStringArray(filePath);


        tagArray = storeTags(textAsCharArray); // creates an array list of tags
```

```
        String tags[] = tagArray.toArray(new String[tagArray.size()]); //creates
a normal string array from tagArray


        int errorLine1; // initialized for 1st rule check
        int errorLine2 = 0; // initialized for 2nd rule check


        // gets the error line of rule 1 check
        errorLine1 = rule1Check(textAsStringArray);
        stack = rule234Check(tags); // returns the stack from the rule check


        // gets the error line if the stack is not empty
        if(!stack.isEmpty()){
          errorLine2 = getLineForRule2(textAsStringArray, stack.getTop());
        }


        // prints output if no errors
        if(errorLine1 == 0 && stack.isEmpty()){
          System.out.println("Congratulations...");
          System.out.println("The given HTML file meets all the tag rules..");
        }


        // Prints error information
        else{
          System.out.println("Oops... There is a problem..");
          if(errorLine1 != 0){
            System.out.println("One of the tags on line " + errorLine1 + " is
missing a < or >");
          }
          if(errorLine2 != 0){
            System.out.println("The " + stack.getTop() + " tag at line #" +
errorLine2 + " does not meet the tag rules..");
```

```
        }


      }


  }


  /*
   * Returns the line that an error occurred
   *
   * @param text: input file with each element corresponding to a line from
file
   * @param top: the top of the stack
   * @return errorLine2: the line where the error occurred
   */
  public static int getLineForRule2(String[] text, String top){
    int errorLine2 = 0;
    for(int i = 0; i < text.length; i ++){
      if (text[i].contains(top)){
        // System.out.println(text[i]);
        errorLine2 = i + 1;
        return errorLine2;
      }
    }
    return 0;
  }


  /*
   * Checks to make sure all tags are enclosed in <> using a stack
```

```
     *
     * @param text: input file with each element corresponding to a line from
file
     * @return lineCount: The first line at which a tag was not enclose in <>
     */
    public static int rule1Check(String[] text)throws Exception{
      Stack<Character> stack = new Stack<>();
      int lineCount = 1;
      for(String line : text){
        if(line != null){
          char charArray[] = line.toCharArray();
            for(char ch : charArray){
              if(ch == '<'){
                stack.push(ch);
              }
            }
            for(char ch : charArray){
              if(ch == '>'){
                stack.pop();
              }
            }
          if(!stack.isEmpty()){
            return lineCount;
          }
          lineCount += 1;
        }

      }
      return 0;

    }
```

```
    /*
     * Checks for rules 2, 3, and 4 using a stack
     *
     * @param tagArray: Array of all the tags in the file
     * @return stack: The stack used to check the rules. Will be empty if all
rules passed
     */
    public static Stack<String> rule234Check(String[] tagArray){
      Stack<String> stack = new Stack<>();
      for(String element : tagArray){
        char elementChar[] = element.toCharArray();
        if(!element.contains("/") && elementChar[1] != '!' &&
!element.equals("<br>") && !element.equals("<hr>")){
          stack.push(element);
          // System.out.println("Pushing: " + element + " endtag: " +
getEndTag(element));
        }
        else if(element.equals(getEndTag(stack.getTop()))){
          String delElement = stack.pop();
          // System.out.println("Popping: " + delElement);
        }
      }
      if(stack.isEmpty()){
        // System.out.println("Rule 2 passed");
        return stack;
      }
      // System.out.println("Rule 2 failed");
      return stack;

    }
```

```
/*
 *  Returns the endtag of a given tag
 *
 * @param tag: the start tag
 * @return endTag: the corresponding end tag for the given tag
 */
public static String getEndTag(String tag){
  if(tag == null){
    return null;
  }
  String commonTags[] = {"html", "head", "title", "body", "center"};
  String endTag;
  char tagChar[] = tag.toCharArray();


  for(String element : commonTags){
    if(tag.contains(element)){
      endTag = "</" + element + '>';
      return endTag;
    }
  }

  if(tagChar.length == 3){
    endTag = "</" + tagChar[1] + '>';
    return endTag;
  }

  if(tagChar[1] == 'h'){
    endTag = "</" + tagChar[1] + tagChar[2] + '>';
    return endTag;
```

```
  }

  if(tagChar[1] == 'a'){
    endTag = "</" + 'a' + '>';
    return endTag;
  }

  return null;


}


/*
 * Takes the char array of the file and returns an array of all tags
 *
 * @param textAsArray: the input file as a character array
 * @return tagArray: an array of all tags in the file
 */
public static ArrayList<String> storeTags(char[] textAsArray){
  int length = textAsArray.length;
  String tag;
  int begin = 0;
  int end = 0;
  ArrayList<String> tagArray = new ArrayList<>();

  for(int i = 0; i < length; i++){
    if(textAsArray[i] == '<'){
      begin = i;
    }
```

```
    else if(textAsArray[i] == '>'){
      end = i;
      tag = getTag(textAsArray, begin, end);
      tagArray.add(tag);
    }
  }

  return tagArray;
}




/*
 * Gets a specific tag by the indes of '<' and '>''
 *
 * @param textAsArray: the input file as a char array
 * @param begin: the index of the '<'
 * @paraam end: the index of the '>'
 * @return tag.toString(): the tag
 */
public static String getTag(char[] textAsArray, int begin, int end){
  StringBuilder tag = new StringBuilder();
  for(int start = begin;  start <= end; start++){
    if(textAsArray[start] != '\r'){
      tag.append(textAsArray[start]);
    }

  }
```

```
    return tag.toString();

  }

  /*
   * Produces a char array from the input file
   *
   * @param filePath: the file path of the .txt file
   * @return textFileAsArray: the .txt file as a character array
   */
  public static char[] fileToArray(String filePath)throws Exception{
  File file = new File(filePath);
  FileReader fr = new FileReader(file);
  BufferedReader br = new BufferedReader(fr);
  int k = 0;
  int approxLength = (int)(file.length() + 10);
  char[] textFileAsArray = new char[approxLength];

  while(br.ready()){
    textFileAsArray[k] = (char)br.read();
    // System.out.print(textFileAsArray[k]);

    k++;
  }
  br.close();

  return textFileAsArray;

}
  /*
```

```
     * Produces a string array from the input file
     * each element in the array represents a line from the file
     *
     * @param filePath: the file path of the .txt file
     * @return textFileAsArray: the .txt file as a string array
     */
    public static String[] fileToStringArray(String filePath)throws Exception{
    File file = new File(filePath);
    FileReader fr = new FileReader(file);
    BufferedReader br = new BufferedReader(fr);
    int k = 0;
    int approxLength = (int)(file.length() + 10);
    String[] textFileAsArray = new String[approxLength];

    while(br.ready()){
      textFileAsArray[k] = br.readLine();
      // System.out.print(textFileAsArray[k]);

      k++;
    }
    br.close();

    return textFileAsArray;

  }
}
```