```java
package A4P1;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.LinkedList;
import java.util.Scanner;

public class HorspoolMapMcCoy {

    public static void main(String[] args) throws FileNotFoundException {
        if (args.length != 2) { // if there is not 2 files inputed
            System.out.print("Please input two files");
            System.exit(0);

        } else {
            // Instantiate object
            new HorspoolMapMcCoy(args);
        }
    }
// End of main Method

    public HorspoolMapMcCoy(String[] args) throws FileNotFoundException {

        Scanner Sc = null;
        Sc = new Scanner(new File(args[0]));

        PrintWriter w = new PrintWriter(new File(args[1]));
        // while there is more to read in the input file
        while (Sc.hasNextLine()) {
            System.out.println("new input reached ");
            // the first line in the input is the text
            String text = Sc.nextLine();

            // the second line is the pattern we are searching for within the text
            String pattern = Sc.nextLine();
            System.out.println(text);
            System.out.println(pattern);
            // create a linked list of character
            LinkedList bmt = new LinkedList();

            // if the pattern is longer than the text print that there isnt a match because
            // its impossible
            if (pattern.length() > text.length()) {
                w.println("Pattern > Text , no match for pattern " + pattern);
                // Sc.nextLine();
            }
            // else if the pattern is not longer than the text
            else {
                Sc.nextLine();

                // for every character of the pattern
                for (int i = 0; i < pattern.length(); i++) {
                    // if the bmt doesn't have the character
                    if (!(bmt.contains(pattern.charAt(i)))) {
                        // if the BMT doesnt have the character and it is the last one make the
value to
```

```
                    // the size of the pattern
                    if (i == pattern.length() - 1) {
                        bmt.add(pattern.charAt(i));
                        bmt.add(bmt.indexOf(pattern.charAt(i)) + 1, pattern.length());

                    }
                    // if it is only not already in the bmt add it
                    else {
                        bmt.add(pattern.charAt(i));
                        bmt.add(0);


                    }
                }
                if (i == pattern.length() - 1)
                    ;
                else
                    // set its bmt value to the right of it
                    bmt.set(bmt.indexOf(pattern.charAt(i)) + 1, (pattern.length() - i - 1));
            }

            // add a wildcard to the end
            bmt.add('*');
            // add the length as the value of the wildcard to the right of it
            bmt.add(bmt.indexOf('*') + 1, pattern.length());
            System.out.print(bmt);

            boolean matchFound = false;
            int textIndex = (pattern.length() - 1);
            while (!matchFound || textIndex == text.length()) {
                int patternIndex = (pattern.length() - 1);

                System.out.println("textindex: " + textIndex);
                System.out.println("patternIdnex: " + patternIndex);
                if (pattern.charAt(patternIndex) == text.charAt(textIndex)) {
                    while (pattern.charAt(patternIndex) == text.charAt(textIndex)) {
                        // if they are the same go back one to compare the previouse
characters
                        patternIndex--;
                        textIndex--;
                        // if you make it to the first location of the pattern with them all
equaling
                        // then you have found the match
                        if (patternIndex == 0) {
                            matchFound = true;
                            System.out.println("match found");
                            break;
                        }
                    }
                }
                // if the locations do not match
                else {
                    // if the bmt has the character
                    if (bmt.contains(text.charAt(textIndex)))
                        // move to the left the amount
                        textIndex = textIndex + (int)
bmt.get(bmt.indexOf(text.charAt(textIndex)) + 1);
                    // if it doesn't use the wildcard
```

```
                    else
                        textIndex = textIndex + (int) bmt.get(bmt.indexOf('*') + 1);
                }
            }
            w.println("text: " + text + "\t" + "pattern: " + pattern);
            w.print("bmt Table: {");

            // go through bmt printing each character along with their values
            for (int i = 0; i < bmt.size(); i = i + 2) {
                // if its not the last one use a comma
                if (!(i == bmt.size() - 2))
                    w.print(bmt.get(i) + "=" + bmt.get(i + 1) + ", ");
                else
                    // if it is the last one do not use a comma
                    w.print(bmt.get(i) + "=" + bmt.get(i + 1));
            }
            w.println("}");
            if (matchFound)
                w.println("Found a match for pattern " + pattern + " at pos " + textIndex);
            else
                w.println("No match found for pattern " + pattern + " in text");
            w.println();
        }
    }
    w.close();
    Sc.close();
}// end constructor
}
```