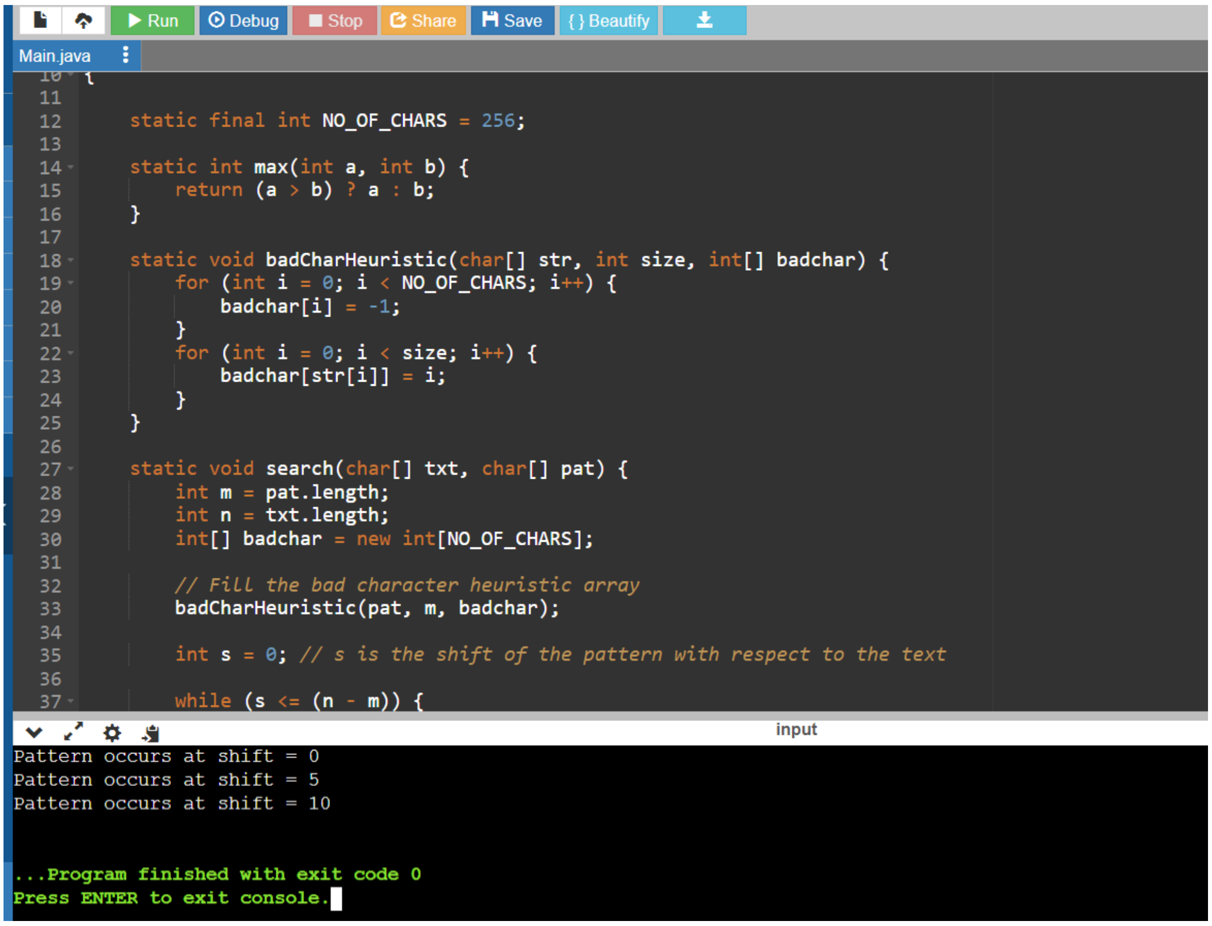
BOYERMOOREIMPLEMENTATION

```java
package com.wipro;

public class BoyerMooreImplementation {

    static final int NO_OF_CHARS = 256;

    static int max(int a, int b) {
        return (a > b) ? a : b;
    }

    static void badCharHeuristic(char[] str, int size, int[] badchar) {
        for (int i = 0; i < NO_OF_CHARS; i++) {
            badchar[i] = -1;
        }
        for (int i = 0; i < size; i++) {
            badchar[str[i]] = i;
        }
    }

    static void search(char[] txt, char[] pat) {
        int m = pat.length;
        int n = txt.length;
        int[] badchar = new int[NO_OF_CHARS];

        // Fill the bad character heuristic array
        badCharHeuristic(pat, m, badchar);

        int s = 0; // s is the shift of the pattern with respect to the text

        while (s <= (n - m)) {
            int j = m - 1;

            // Decrease index j of pattern while characters of pattern and text
are matching
            while (j >= 0 && pat[j] == txt[s + j]) {
                j--;
            }

            // If the pattern is present at the current shift, then index j will
become -1
            if (j < 0) {
                System.out.println("Pattern occurs at shift = " + s);

                // Shift the pattern so that the next character in text aligns
with the last occurrence of it in the pattern.
                // The condition s + m < n is necessary for the case when pattern
occurs at the end of text
                s += (s + m < n) ? m - badchar[txt[s + m]] : 1;
            } else {
                // Shift the pattern so that the bad character in text aligns with
the last occurrence of it in pattern.
                // The max function is used to make sure that we get a positive
shift.
                s += max(1, j - badchar[txt[s + j]]);
                }
            }
        }

        public static void main(String[] args) {
            // Text in which pattern occurs
            char[] txt = "123651266512".toCharArray();
            // Pattern to search
            char[] pat = "12".toCharArray();
            search(txt, pat);
        }
}
```



```java
{

    static final int NO_OF_CHARS = 256;

    static int max(int a, int b) {
        return (a > b) ? a : b;
    }

    static void badCharHeuristic(char[] str, int size, int[] badchar) {
        for (int i = 0; i < NO_OF_CHARS; i++) {
            badchar[i] = -1;
        }
        for (int i = 0; i < size; i++) {
            badchar[str[i]] = i;
        }
    }

    static void search(char[] txt, char[] pat) {
        int m = pat.length;
        int n = txt.length;
        int[] badchar = new int[NO_OF_CHARS];

        // Fill the bad character heuristic array
        badCharHeuristic(pat, m, badchar);

        int s = 0; // s is the shift of the pattern with respect to the text

        while (s <= (n - m)) {
```

```
Pattern occurs at shift = 0
Pattern occurs at shift = 5
Pattern occurs at shift = 10


...Program finished with exit code 0
Press ENTER to exit console.
```