Task 2:

Rabin-Karp Substring Search

Implement the Rabin-Karp algorithm for substring search using a rolling hash. Discuss the impact of hash collisions on the algorithm's performance and how to handle them.

ANS:

```
package Day17Tasks;

public class RabinKarp {

    // Prime number for the hash function
    private static final int PRIME = 101;

    // Rabin-Karp search algorithm
    public static void rabinKarpSearch(String text, String pattern) {
        int textLength = text.length();
        int patternLength = pattern.length();

        if (patternLength > textLength) {
            System.out.println("Pattern length is greater than text length. No match
possible.");
            return;
        }

        long patternHash = createHash(pattern, patternLength);
        long textHash = createHash(text, patternLength);

        for (int i = 0; i <= textLength - patternLength; i++) {
            if (patternHash == textHash && checkEqual(text, i, pattern)) {
                System.out.println("Pattern found at index " + i);
            }
            if (i < textLength - patternLength) {
                textHash = recalculateHash(text, i, i + patternLength, textHash,
patternLength);
            }
        }
    }

    // Method to create the initial hash value
    private static long createHash(String str, int length) {
        long hash = 0;
        for (int i = 0; i < length; i++) {
            hash += str.charAt(i) * Math.pow(PRIME, i);
        }
        return hash;
    }

    // Method to recalculate hash value for the next window
```

```
    private static long recalculateHash(String str, int oldIndex, int newIndex, long
oldHash, int patternLength) {
        long newHash = oldHash - str.charAt(oldIndex);
        newHash /= PRIME;
        newHash += str.charAt(newIndex) * Math.pow(PRIME, patternLength - 1);
        return newHash;
    }

    // Method to check if the current substring matches the pattern
    private static boolean checkEqual(String text, int start, String pattern) {
        for (int i = 0; i < pattern.length(); i++) {
            if (text.charAt(start + i) != pattern.charAt(i)) {
                return false;
            }
        }
        return true;
    }

    public static void main(String[] args) {
        String text = "ABABDABACDABABCABAB";
        String pattern = "ABABCABAB";
        rabinKarpSearch(text, pattern);
    }
}
```

OUTPUT:

Pattern found at index 10