Kodowanie kanałowe

Wykonaj w formie programistycznej implementacji poniżej przedstawione zadania.

1) Zaimplementuj funkcję kodującą kodem Hamming (7,4) zadany strumień binarny. Do generowania strumienia binarnego użyj funkcji **S2BS** napisanej na laboratoriach „5. Modulacja dyskretna”.

```
public int[][] hamming74Encode(String text) {
    int len = s2BS.stringToBinaryStream(text, isBigEndian: false).length();
    SimpleMatrix bit4 = make4bit(text, len);
    SimpleMatrix G = getSimpleMatrix( isG: true, row: 4, col: 7);
    int[][] scores = new int[len / 4][7];
    for (int k = 0; k < len / 4; k++) {
        for (int i = 0; i < G.numCols(); i++) {
            double sum = 0;
            for (int j = 0; j < G.numRows(); j++) {
                sum += bit4.get(k, j) * G.get(j, i);
            }
            scores[k][i] = ((int) sum % 2);
        }
    }
    return scores;
}
```

2) Napisz funkcję negującą wskazany numer bitu w strumieniu binarnym z zadania pierwszego.

```
private int[][] noise(int[][] array, int row, int col, int len) {
    int[][] tmp = new int[len / 4][7];

    for (int i = 0; i < len / 4; i++) {
        for (int j = 0; j < 7; j++) {
            if (i == row && j == col) {
                if (array[i][j] == 0) {
                    tmp[i][j] = 1;
                } else {
                    tmp[i][j] = 0;
                }
            } else {
                tmp[i][j] = array[i][j];
            }
        }

    }
    return tmp;
}
```

3) Zaimplementuj funkcję dekodującą kod Hamminga (7,4), sprawdź poprawność działania.

```
public int[][] hammingDetect(int[][] array, int len) {
    int[][] scores = new int[len / 4][3];

    for (int k = 0; k < len / 4; k++) {
        for (int i = 0; i < 3; i++) {
            double sum = 0;
            for (int j = 0; j < 7; j++) {
                sum += array[k][j] * h[j][i];
            }
            scores[k][i] = (int) sum % 2;
        }
    }
    return scores;
}
```

```java
public int[][] hammingDecode(int[][] detected, int[][] text, int len) {
    int[][] scores = new int[len / 4][4];
    boolean flag = false;
    int index = 0;
    for (int k = 0; k < len / 4; k++) {
        for (int i = 0; i < 7; i++) {
            if ((detected[k][0] == h[i][0]) && (detected[k][1] == h[i][1]) && (detected[k][2] == h[i][2])) {
                flag = true;
                index = i;
            }
            if (flag) {
                if (text[k][index] == 1) {
                    text[k][index] = 0;
                } else {
                    text[k][index] = 1;
                }
            }
            flag = false;
        }
        scores = removeRedundancyBits(text, len);
    }
    return scores;
}
```

Wynik:

```
Text:01010000
Hamming: 01010100000000
Noise: 01010101000000
Detected: 000011
Decoded: 01010000
```