Name : Prem Vinod Bansod
Roll No : 41310
Assignment No : 02 (ICS)

Code:

```
import java.util.*;

public class AES {

	static int key[];
	static int plaintext[];
	static int round_key1[], round_key2[];
	static int subNibbles[][] =    {{9,4,10,11}, {13,1,8,5}, {6,2,0,3},
{12,14,15,7}};
	static int invSubNibbles[][] = {{10,5,9,11}, {1,7,8,15}, {6,0,2,3},
{12,4,13,14}};
	static int RC1[] = {1,0,0,0,0,0,0,0};
	static int RC2[] = {0,0,1,1,0,0,0,0};
	static int lookupTable [][] = {{0,2,4,6,8,10,12,14,3,1,7,5,11,9,15,13},

{0,4,8,12,3,7,11,15,6,2,14,10,5,1,13,9},

{0,9,1,8,2,11,3,10,4,13,5,12,6,15,7,14}};

	static int[] xor(int[] input1, int input2[]) {
		int output[] = new int[input1.length];
	   for (int i = 0; i < input1.length; i++)
	      output[i] = input1[i] ^ input2[i];

	   return output;
	}


	static int getDecimal(int input[]) {
	int output = 0;
	int power =  0;
	for(int i = input.length - 1;i >=0;i--) {
		output += input[i] * (int) Math.pow(2,power);
```

```
            power++;
        }
        return output;
    }


        static int[] getBinary(int decimal) {
        int output[] = new int[4];
        int j = 3 ;
        while(decimal > 0) {
            output[j] = decimal % 2;
            decimal = decimal / 2;
            j--;
        }
        return output;
    }


        static String getHexaDecimal(int input[]) {
            String output = "";
            int decimal = getDecimal(input);
            if(decimal > 9) {
                String hexa[] = {"A","B","C","D","E","F"};
                int index = decimal - 10;
                output = hexa[index];
            }
            else
                output = String.valueOf(decimal);

            return output;
        }


        static void display(int input[]) {
            for(int i = 0;i<input.length;i++) {
                if(i % 4 == 0 && i != 0)
                    System.out.print(" ");
                System.out.print(input[i]);
            }
```

```
        System.out.println();
    }

    static void displayHex(int input[]) {
        for(int i = 0;i<input.length;i=i+4)

    System.out.print(getHexaDecimal(Arrays.copyOfRange(input, i,
i+4)));
        System.out.println();
    }

    static int[] rotateNibbles(int word[]) {
        int output[] = Arrays.copyOfRange(word, 0, word.length);
        for(int i = 0; i < 4 ;i++) {
            int temp = output[i];
            output[i] = output[i+4];
            output[i+4] = temp;
        }
        return output;
    }


    static int[] shiftRows(int input[]) {
        int output[] = Arrays.copyOfRange(input, 0, input.length);

        for(int i=4;i<8;i++) {
            int temp = output[i];
            output[i] = output[i+8];
            output[i+8] = temp;
        }
        return output;
    }

    static int[] subNibblies(int input[]) {

        int output[] = new int[4];
        int row = getDecimal(Arrays.copyOfRange(input, 0, 2));
        int col = getDecimal(Arrays.copyOfRange(input, 2, 4));
```

```
        int substitute = subNibbles[row][col];
        output = getBinary(substitute);
        return output;
    }

    static int[] invSubNibblies(int input[]) {

        int output[] = new int[4];
        int row = getDecimal(Arrays.copyOfRange(input, 0, 2));
        int col = getDecimal(Arrays.copyOfRange(input, 2, 4));

        int substitute = invSubNibbles[row][col];
        output = getBinary(substitute);
        return output;
    }

    static int[] append(int array1[],int array2[]) {
        int output[] = new int[array1.length + array2.length];
        for(int i=0; i<array1.length; i++)
            output[i] = array1[i];

        for(int i=array1.length; i<array1.length + array2.length; i++)
            output[i] = array2[i - array1.length];
        return output;
    }


    static int[] append4nibbles(int S00[],int S10[], int S01[], int S11[]) {

        int left_half[]  = append(S00, S10);
        int right_half[] = append(S01, S11);
        return append(left_half, right_half);
    }


    static int[] performMixColumnsMultiplication(int S00[],int S10[], int
S01[], int S11[]) {
```

```
            int temp[] = getBinary(lookupTable[1][getDecimal(S10)]);
            int new_S00[] = xor(S00,temp);

            temp = getBinary(lookupTable[1][getDecimal(S11)]);
            int new_S01[] = xor (S01,temp);

            temp = getBinary(lookupTable[1][getDecimal(S00)]);
            int new_S10[] = xor(S10,temp);

            temp = getBinary(lookupTable[1][getDecimal(S01)]);
            int new_S11[] = xor(S11,temp);

            return append4nibbles(new_S00, new_S10, new_S01,
new_S11);

        }

        static int[] performInvMixColumnsMultiplication(int S00[],int S10[],
int S01[], int S11[]) {

            int temp1[] = getBinary(lookupTable[2][getDecimal(S00)]);
            int temp2[] = getBinary(lookupTable[0][getDecimal(S10)]);
            int new_S00[] = xor(temp1,temp2);

            temp1 = getBinary(lookupTable[2][getDecimal(S01)]);
            temp2 = getBinary(lookupTable[0][getDecimal(S11)]);
            int new_S01[] = xor (temp1,temp2);

            temp1 = getBinary(lookupTable[0][getDecimal(S00)]);
            temp2 = getBinary(lookupTable[2][getDecimal(S10)]);
            int new_S10[] = xor(temp1,temp2);

            temp1 = getBinary(lookupTable[0][getDecimal(S01)]);
            temp2 = getBinary(lookupTable[2][getDecimal(S11)]);
            int new_S11[] = xor(temp1,temp2);

            return append4nibbles(new_S00, new_S10, new_S01,
new_S11);
        }
```

```
    static void keyGeneration() {
        System.out.println("\n\nKey Generation Process Started");
        System.out.println("Three Keys for encryption are:");
        round_key1 = new int [16];
        round_key2 = new int [16];

        int w0[] = Arrays.copyOfRange(key, 0, 8);
        int w1[] = Arrays.copyOfRange(key, 8, 16);
        int w2[] = new int[8];
        int w3[] = new int[8];
        int w4[] = new int[8];
        int w5[] = new int[8];

        // ROUND1 KEY GENERATION
        int temp1[] = rotateNibbles(w1);
        int left_part[]  = subNibblies(Arrays.copyOfRange(temp1, 0,
4));
        int right_part[] = subNibblies(Arrays.copyOfRange(temp1, 4,
8));
        int temp2[] = append(left_part,right_part);
        int t1[] = xor(temp2,RC1);
        w2 = xor(w0,t1);
        w3 = xor(w2, w1);
        round_key1 = append(w2, w3);

        // ROUND2 KEY GENERATION
        temp1 = rotateNibbles(w3);
        left_part  = subNibblies(Arrays.copyOfRange(temp1, 0, 4));
        right_part = subNibblies(Arrays.copyOfRange(temp1, 4, 8));
        temp2 = append(left_part,right_part);
        int t2[] = xor(temp2,RC2);
        w4 = xor(w2,t2);
        w5 = xor(w4, w3);
        round_key2 = append(w4, w5);
```

```
        System.out.print("Key 0:");
        display(key);
        System.out.print("Key 1:");
        display(round_key1);
        System.out.print("Key 2:");
        display(round_key2);
    }


    static int[] encrypt() {
        System.out.println("\n\nEncryption Process Started");

        // PreRound
        int encoding[] = xor(key,plaintext);
        System.out.print("Output Of Round 0: ");
        display(encoding);

        // Round 1
        int S00[] = Arrays.copyOfRange(encoding, 0, 4);
        int S10[] = Arrays.copyOfRange(encoding, 4, 8);
        int S01[] = Arrays.copyOfRange(encoding, 8, 12);
        int S11[] = Arrays.copyOfRange(encoding, 12, 16);

        S00  = subNibblies(S00);
        S10  = subNibblies(S10);
        S01  =  subNibblies(S01);
        S11  = subNibblies(S11);
        encoding = append4nibbles(S00, S10, S01, S11);
        encoding = shiftRows(encoding);
        S00 = Arrays.copyOfRange(encoding, 0, 4);
        S10 = Arrays.copyOfRange(encoding, 4, 8);
        S01 = Arrays.copyOfRange(encoding, 8, 12);
        S11 = Arrays.copyOfRange(encoding, 12, 16);
        encoding = performMixColumnsMultiplication(S00, S10, S01,
S11);
        encoding = xor(encoding, round_key1);
        System.out.print("Output Of Round 1: ");
        display(encoding);
```

```
        //Round 2
        S00 = Arrays.copyOfRange(encoding, 0, 4);
        S10 = Arrays.copyOfRange(encoding, 4, 8);
        S01 = Arrays.copyOfRange(encoding, 8, 12);
        S11 = Arrays.copyOfRange(encoding, 12, 16);

        S00  = subNibblies(S00);
        S10  = subNibblies(S10);
        S01  =  subNibblies(S01);
        S11  = subNibblies(S11);

        encoding = append4nibbles(S00, S10, S01, S11);
        encoding = shiftRows(encoding);
        encoding = xor(encoding, round_key2);
        System.out.print("Final Ciper Text: ");
        display(encoding);



        return encoding;
}

static void decrypt(int cipher[]) {
        System.out.println("\n\nDecryption Process Started");

        // PreRound
        int decoding[] = xor(round_key2,cipher);
        System.out.print("Output Of Round 0: ");
        display(decoding);


        //Round 1
        decoding = shiftRows(decoding);
        int S00[] = Arrays.copyOfRange(decoding, 0, 4);
        int S10[] = Arrays.copyOfRange(decoding, 4, 8);
        int S01[] = Arrays.copyOfRange(decoding, 8, 12);
        int S11[] = Arrays.copyOfRange(decoding, 12, 16);

        S00  = invSubNibblies(S00);
```

```
            S10  = invSubNibblies(S10);
            S01  = invSubNibblies(S01);
            S11  = invSubNibblies(S11);
            decoding = append4nibbles(S00, S10, S01, S11);
            decoding = xor(decoding, round_key1);
            System.out.print("Output Of Round 1: ");
            display(decoding);

            //Round2
            S00 = Arrays.copyOfRange(decoding, 0, 4);
            S10 = Arrays.copyOfRange(decoding, 4, 8);
            S01 = Arrays.copyOfRange(decoding, 8, 12);
            S11 = Arrays.copyOfRange(decoding, 12, 16);
            decoding = performInvMixColumnsMultiplication(S00, S10,
S01, S11);
            decoding = shiftRows(decoding);
            S00 = Arrays.copyOfRange(decoding, 0, 4);
            S10 = Arrays.copyOfRange(decoding, 4, 8);
            S01 = Arrays.copyOfRange(decoding, 8, 12);
            S11 = Arrays.copyOfRange(decoding, 12, 16);
            S00  = invSubNibblies(S00);
            S10  = invSubNibblies(S10);
            S01  = invSubNibblies(S01);
            S11  = invSubNibblies(S11);
            decoding = append4nibbles(S00, S10, S01, S11);
            decoding = xor(decoding, key);
            System.out.print("Original Text: ");
            display(decoding);

        }


        public static void main(String[] args) {
            // TODO Auto-generated method stub
            key = new int[16];
            plaintext = new int[16];
            Scanner sc = new Scanner(System.in);

            System.out.println("Enter 16 bit key");
```

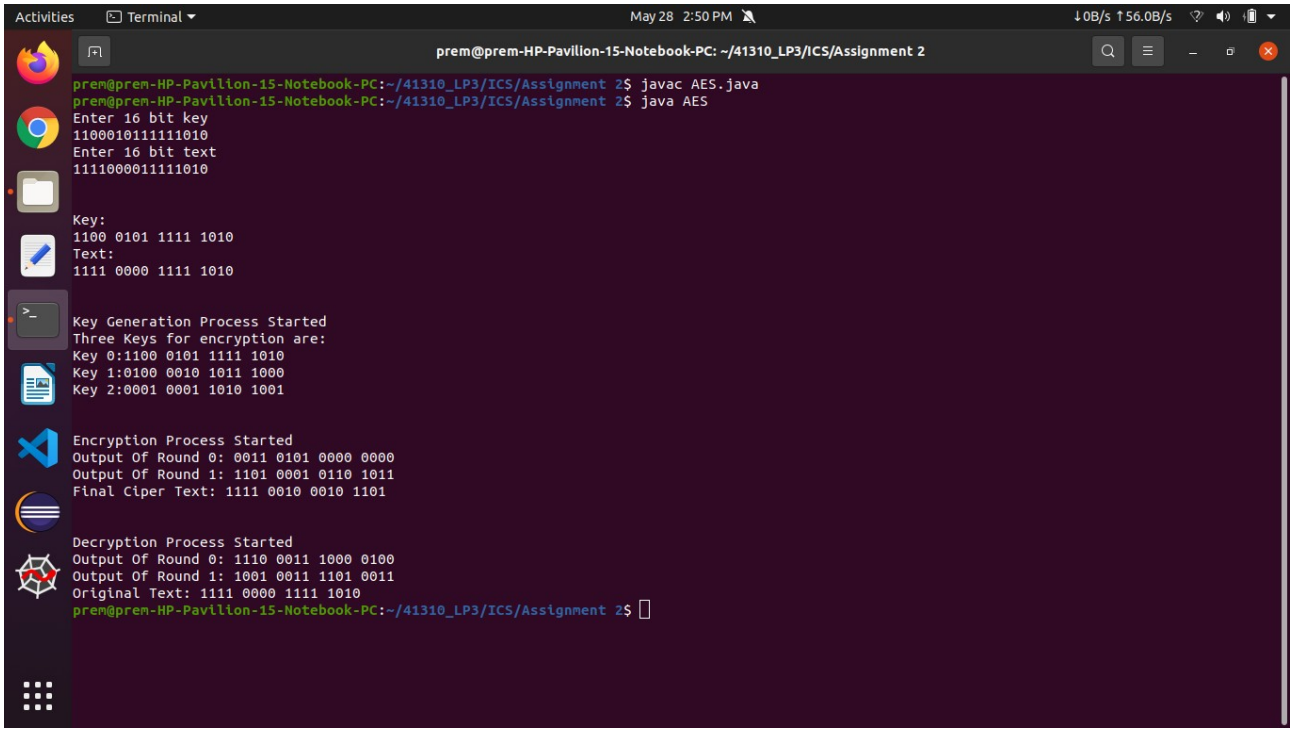
```
            String keyString = sc.next();
            for(int i=0;i<16;i++)
                key[i] =
Integer.parseInt(String.valueOf(keyString.charAt(i)));


            System.out.println("Enter 16 bit text");
            String plaintextString = sc.next();
            for(int i=0;i<16;i++)
                plaintext[i] =
Integer.parseInt(String.valueOf(plaintextString.charAt(i)));

            System.out.println();
            System.out.println();
            System.out.println("Key: ");
            display(key);
            System.out.println("Text:");
            display(plaintext);
            keyGeneration();
            int cipher[] = encrypt();
            decrypt(cipher);
            sc.close();
        }

}
```

output :


Activities
Terminal
May 28 2:50 PM
0B/s 56.0B/s
prem@prem-HP-Pavilion-15-Notebook-PC: ~/41310_LP3/ICS/Assignment 2
prem@prem-HP-Pavilion-15-Notebook-PC:~/41310_LP3/ICS/Assignment 2$ javac AES.java
prem@prem-HP-Pavilion-15-Notebook-PC:~/41310_LP3/ICS/Assignment 2$ java AES
Enter 16 bit key
1100010111111010
Enter 16 bit text
1111000011111010
Key:
1100 0101 1111 1010
Text:
1111 0000 1111 1010
Key Generation Process Started
Three Keys for encryption are:
Key 0:1100 0101 1111 1010
Key 1:0100 0010 1011 1000
Key 2:0001 0001 1010 1001
Encryption Process Started
Output Of Round 0: 0011 0101 0000 0000
Output Of Round 1: 1101 0001 0110 1011
Final Ciper Text: 1111 0010 0010 1101
Decryption Process Started
Output Of Round 0: 1110 0011 1000 0100
Output Of Round 1: 1001 0011 1101 0011
Original Text: 1111 0000 1111 1010
prem@prem-HP-Pavilion-15-Notebook-PC:~/41310_LP3/ICS/Assignment 2$