In main I created a 2d array that is a binary representation of the alphabet, so each row is 5 bits of one letter. Then I created an array that is a binary representation of the word “CRYPTOGRAPHY” which has 12 letters which means that this word is consisted of 12*5=60 bits in total. Then from the encrypted text in the text file I took the first 60 bits so now I got a pair of a ciphertext and a plaintext and I can figure out the output of the Geffe generator (the first 60 bits).

We know that in ¾ of the cases the Geffe generator values overlap with the values that the LFSR1 and LFSR3 produce separately, so I used that to figure out the correct initial keys for these two registers. Then after I find those 2 keys, I go through all (actually until I find the correct one) possible initial keys for LFSR2 and search for the one that gives us 100% overlapping of the 60 bits that we already have for the Geffe and 60 bits that we will generate with Geffe but with the initial keys that we found for the three LFSR’s.

**How I find the keys for each LFSR:**

For each LFSR I generated a 2D array that stores all of its possible initial keys and a 1D array of the coefficients in the corresponding characteristic polynomial (I will denote it with p(x) here). The keys were generated based on permutation logic and their length is the order of the polynomial (with this I mean the biggest degree that appears in it – let’s say that that number is m) for the specific register.

<u>The procedure for finding the correct initial keys for LFSR1 and LFSR3 is the same:</u>
I go through all possibilities for the keys and for each key I generate a 60 bit long array which represents the output of that register. I calculate the values in that array with using the previous m bits and the coefficients in the polynomial p(x) with the formula that we know (this is implemented in the function “calculate”):

$$z_j = \sum_{i=1}^{m} c_i z_{j-i} \quad \text{za} \quad j \geq m$$

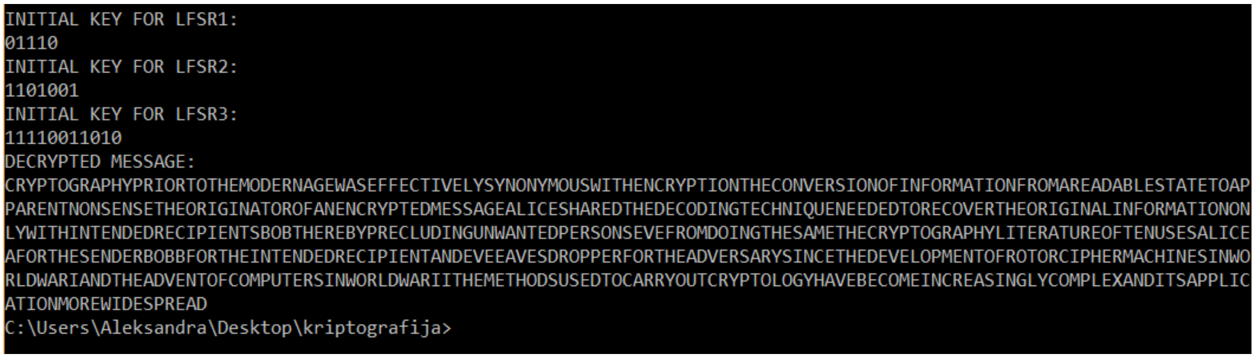
(Ci-a coefficient, Zj-i -a bit )

While doing this I also take count of the number of bits that overlap in the two 60bit arrays (the variable “counter”) and also the index of the key that currently has the biggest percentage of overlapping(“indexofmax”), and in the end my final result a.k.a. the initial key that we want, is the key that it is in the row with index “indexofmax” in the 2D table of keys for the specific register. As I said before, that percentage has to be more than 75% (variable counter to be at least 45). When solving, I printed all of this information for lfsr1, and saw that only one of the keys had an overlap of >=75% (counter was 46), so with this I checked that the program returns the correct key.

<u>Procedure for finding the key for LFSR2:</u>
Now we have the keys for lfsr1 and lfsr3 and we want to find the key fot lfsr2. As I wrote before, when searching for the initial key for lfsr2, we approach it differently, since the overlapping of the values for this register and the Geffe generator is only around 50%. Instead of generating only an output of 60 bits of the lfsr register itself, we additionally calculate the 60 bits that would be the output of the Geffe generator if we have that current 3-tuple of initial keys of the 3 registers. This calculation is implemented in the function “calculateGeffe” and each bit is calculated like this: x1x2+x2x3+x3 (mod2), where each xi is an output bit of the LFSRi. The correct initial key for LFSR2 is the first key in the 2D table of possible keys that gives us a 100% overlap.

After finding the keys, we can decrypt the whole message that we got on the beginning with using the Geffe generator and the three keys for the registers (again with the function “calculateGeffe”). With this we will get the message in bits and we will transform it into a message of letters.

This is the output of my program:



```
INITIAL KEY FOR LFSR1:
01110
INITIAL KEY FOR LFSR2:
1101001
INITIAL KEY FOR LFSR3:
11110011010
DECRYPTED MESSAGE:
CRYPTOGRAPHYPRIORTOTHEMODERNAGEWASEFFECTIVELYSYNONYMOUSWITHENCRYPTIONTHECONVERSIONOFINFORMATIONFROMAREADABLESTATETOAP
PARENTNONSENSETHEORIGINATOROFANENCRYPTEDMESSAGEALICESHAREDTHEDECODINGTECHNIQUENEEDEDTORECOVERTHEORIGINALINFORMATIONON
LYWITHINTENDEDRECIPIENTSBOBTHEREBYPRECLUDINGUNWANTEDPERSONSEVEFROMDOINGTHESAMETHECRYPTOGRAPHYLITERATUREOFTENUSESALICE
AFORTHESENDERBOBBFORTHEINTENDEDRECIPIENTANDEVEEAVESDROPPERFORTHEADVERSARYSINCETHEDEVELOPMENTOFROTORCIPHERMACHINESINWO
RLDWARIANDTHEADVENTOFCOMPUTERSINWORLDWARIITHEMETHODSUSEDTOCARRYOUTCRYPTOLOGYHAVEBECOMEINCREASINGLYCOMPLEXANDITSAPPLIC
ATIONMOREWIDESPREAD
C:\Users\Aleksandra\Desktop\kriptografija>
```