The ciphertext printed below was encrypted using a substitution cipher. Theobjective is to decrypt the ciphertext without knowledge of the key.

a. Provide the relative frequency of all letters A...Z in the ciphertext.

Answer:

Code	Count	Frequency
A	150	13.928%
B	100	9.285%
C	86	7.985%
F	83	7.707%
D	76	7.057%
I	75	6.964%
G	70	6.500%
E	58	5.385%
L	50	4.643%
K	47	4.364%
H	45	4.178%
J	40	3.714%
M	37	3.435%

N	24	2.228%
S	24	2.228%
Q	23	2.136%
O	19	1.764%
P	19	1.764%
U	15	1.393%
R	15	1.393%
V	9	0.836%
T	9	0.836%
Y	3	0.279%

b.

Decrypt the ciphertext with help of the relative letter frequency of the English language (e.g., search Wikipedia for letter frequency analysis). Note that the text is relatively short ; frequencies may not exactly match those listed in Wikipedia or elsewhere.

ELECTRICAL AND COMPUTER ENGINEERS DEVELOP AND CREATE PRODUCTS THAT CHANGE THE WORLD AND MAKE OUR LIVES EASIER THE CELL PHONES WE DEPEND ON THE COMPUTERS USED IN NATIONAL SECURITY AND THE ELECTRICAL SYSTEMS THAT MAKE OUR CARS OPERATE WERE ALL CREATED BY ELECTRICAL AND COMPUTER ENGINEERS AT WPI WE KEEP THAT PROGRESS MOVING FORWARD WITH OUR INNOVATIVE RESEARCH AND OUT-OF-THE BOX APPROACHES THE DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING AT WPI CHALLENGES STUDENTS TO PUSH THEMSELVES TO UNDERSTAND SOCIETYS AND TECHNOLOGYS COMPLEX ISSUES IN A BROADER CONTEXT THAN WHATS IN FRONT OF THEM WE WANT OUR STUDENTS WHETHER THEY ARE EARNING AN UNDERGRADUATE MINOR OR A DOCTORATE TO TACKLE SOCIETYS MOST PRESSING PROBLEMS AND UNCOVER NEW WAYS OF SOLVING THEM WHETHER ITS DEVELOPING SYSTEMS THAT CAN LOCATE FIREFIGHTERS IN THE MIDDLE OF A BURNING BUILDING OR CREATING NEUROPROSTHETICS THAT LOOK AND FUNCTION LIKE NATURAL LIMBS OUR FACULTY AND STUDENTS ARE AT THE FRONT EDGE OF REMARKABLE INNOVATION WHILE ADVANCING TECHNOLOGIES IS AT OUR CORE WE ALSO TAKE HUMAN CONNECTIONS VERY SERIOUSLY IN ECE WE PRIDE OURSELVES ON THE FAMILY-LIKE ATMOSPHERE WE CULTIVATE; FACULTY STUDENTS AND STAFF ENCOURAGE EACH OTHERS EVERY SUCCESS AND ARE THERE FOR THE CHALLENGES BOTH IN THE CLASSROOM AND IN LIFE.

c.

Find the Plaintext/Ciphertext letter pairs, alphabetized by plaintext.

CIPHERTEXT	PLAINTEXT
A	E
B	T
C	A
F	N
D	O
I	R
G	S
E	I
L	C
K	L
H	H
J	D
M	U
N	M
S	P
Q	G
O	W
P	F
U	V
R	Y
V	K
T	B
Y	X

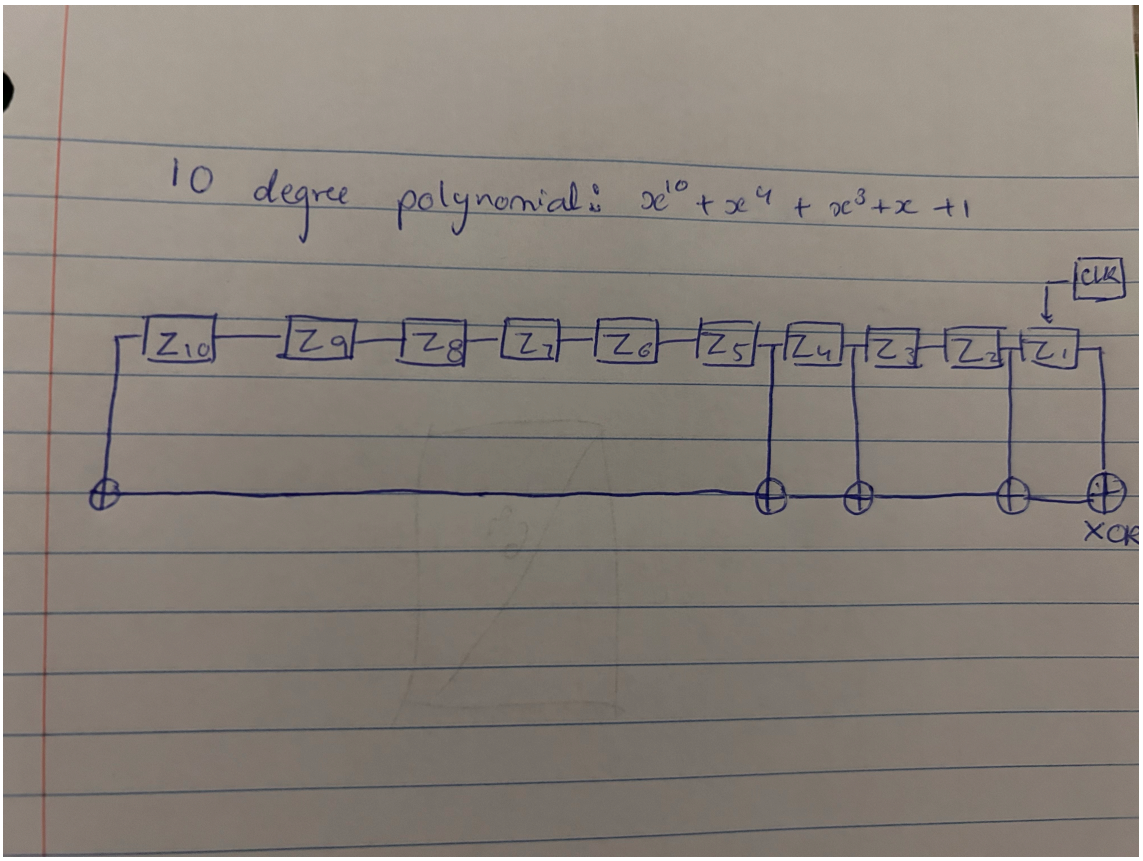
d. Provide letter frequency for the given plain text.

Code	Count	Frequency
E	150	13.928%
T	100	9.285%
A	86	7.985%
N	83	7.707%
O	76	7.057%
R	75	6.964%
S	70	6.500%
I	58	5.385%
C	50	4.643%
L	47	4.364%
H	45	4.178%
D	40	3.714%
U	37	3.435%
M	24	2.228%
P	24	2.228%
G	23	2.136%
W	19	1.764%

F	19	1.764%
V	15	1.393%
Y	15	1.393%
K	9	0.836%
B	9	0.836%
X	3	0.279%

Question 2:

a. Draw a circuit diagram for the given LFSR.



b. Compute the first 512 bits of the output bit stream. You can use any program of your choice.

```
tap_positions = [0, 3, 4, 10]
lfsr_length = 10
initial_state = 0b1010101010
output_stream = []
state = initial_state
period = 0
while period < 512:
    feedback_bit = sum((state >> pos) & 1 for pos in tap_positions) % 2
    output_stream.append(state & 1)
    state = (state >> 1) | (feedback_bit << (lfsr_length - 1))
    period += 1
output_stream_str = ''.join(map(str, output_stream))
print(output_stream_str[:512])
print("Period:", period)
```

Output:
10101011010110111001111100101001111110101100001011011110111111111010
00111011001111001011010101

c. What is the period of the output stream?

Period of output stream is 512

d. Encrypt the following 32 bit plaintext using the first 32 bits of the key stream generated above.
P=`11101100000110111011010011111010`

```
plaintext = "11101100000110111011010011111010"
key_stream = output_stream_str[:32]

# Perform bitwise XOR between plaintext and key stream
ciphertext = ''.join(str(int(p) ^ int(k)) for p, k in zip(plaintext, key_stream))

print("Ciphertext:", ciphertext)
```

Output:

Ciphertext: 10111001011100011001001010100110

e. Decrypt the cipher text you found in part d using the bit same key stream generated above.

```
decrypted_plaintext = ''.join(str(int(c) ^ int(k)) for c, k in zip(ciphertext, key_stream))

print("Decrypted Plaintext:", decrypted_plaintext)
```

Output:

```
Decrypted Plaintext: 11101100000110111011010011111010
```