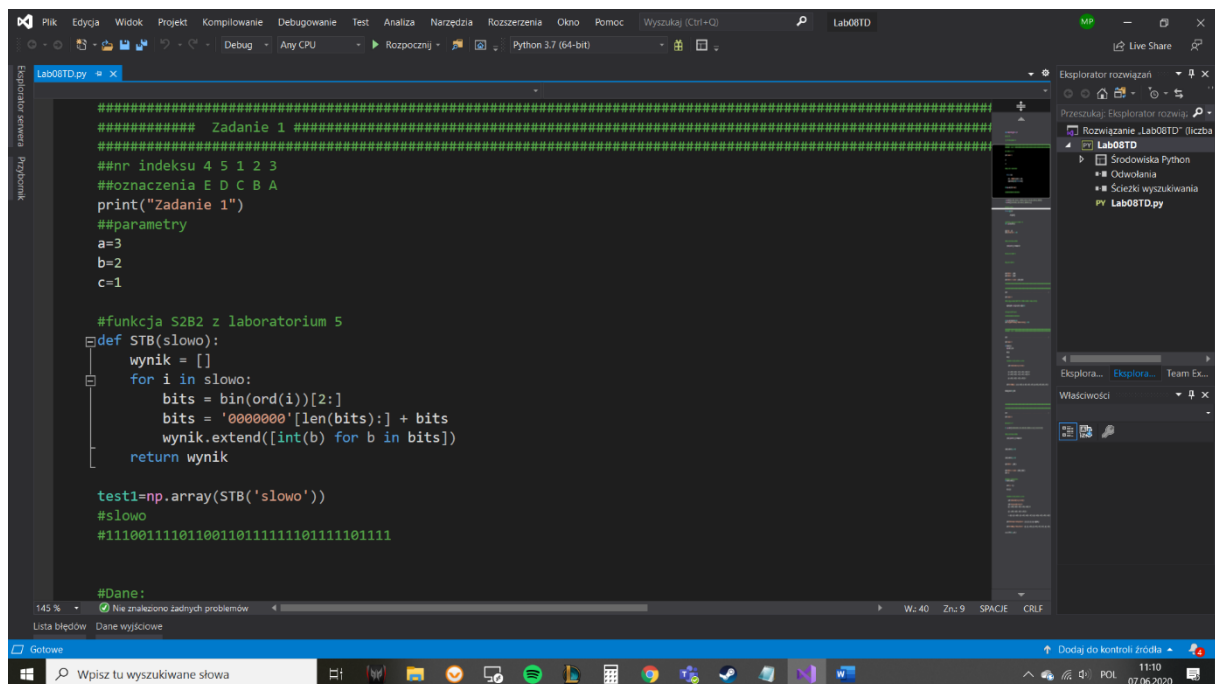


Sprawozdanie Lab08

Mateusz Proc 20 N1 nr 45123



```
##### Zadanie 1 #####
#####
#nr indeksu 4 5 1 2 3
#oznaczenia E D C B A
print("Zadanie 1")
##parametry
a=3
b=2
c=1

#funkcja S2B2 z laboratorium 5
def STB(slowo):
    wynik = []
    for i in slowo:
        bits = bin(ord(i))[2:]
        bits = '0000000'[len(bits):] + bits
        wynik.extend([int(b) for b in bits])
    return wynik

test1=np.array(STB('slowo'))
#slowo
#1111001111011001101111111011110111101111

#Dane:
145 % Nie znaleziono żadnych problemów
Lista błędów Dane wyjściowe
```

Funkcja String To Binary

2/.

The screenshot shows the Visual Studio Code editor with a Python file named 'Lab08TD.py'. The code defines a function 'negation' that takes a binary stream and returns its negation. It also includes test cases for the function. The code is as follows:

```
print("Wektor 1 i 2 łącznie: ", v_tab1, v_tab2)

##### Zadanie 2 #####
print("\n")
print("-----")
print("\n")
print("Zadanie 2")

#funkcja negująca wskazany numer bitu w strumieniu binarnym z zadania pierwszego
def negation(x):
    negation_word = np.logical_not(x).astype(int)
    return negation_word

#test1=np.array('slow')
#slow
#11100111011001101111101111101111

test2=np.array(negation(test1))
print("Zanegowanie wskazanego strumienia binarnego: ", test2)

##### Zadanie 3 #####
```

Funkcja negująca strumień binarny

The screenshot shows the output of the Python script in a terminal window. The output displays the results of the binary stream processing tasks, including the negation of the binary stream and the Hamming code calculation.

```
Zadanie 1
tab: [1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1]
pierwsza część: [1 1 1 0]
druga część: [0 1 1 1]
Wektor 1: [0 0 1 0 1 1 0]
Wektor 2: [0 0 0 1 1 1 1]

Wektor 1 i 2 łącznie: [0 0 1 0 1 1 0] [0 0 0 1 1 1 1]

-----

Zadanie 2
Zanegowanie wskazanego strumienia binarnego: [0 0 0 1 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0]

-----

Zadanie 3
h: [0 0 0]
Znaleziono indeks róny zero
Wektor danych: 1 0 1 1
Kod Hamminga: 1 0 1 1 1 1 0 1 0 1 1 1

-----

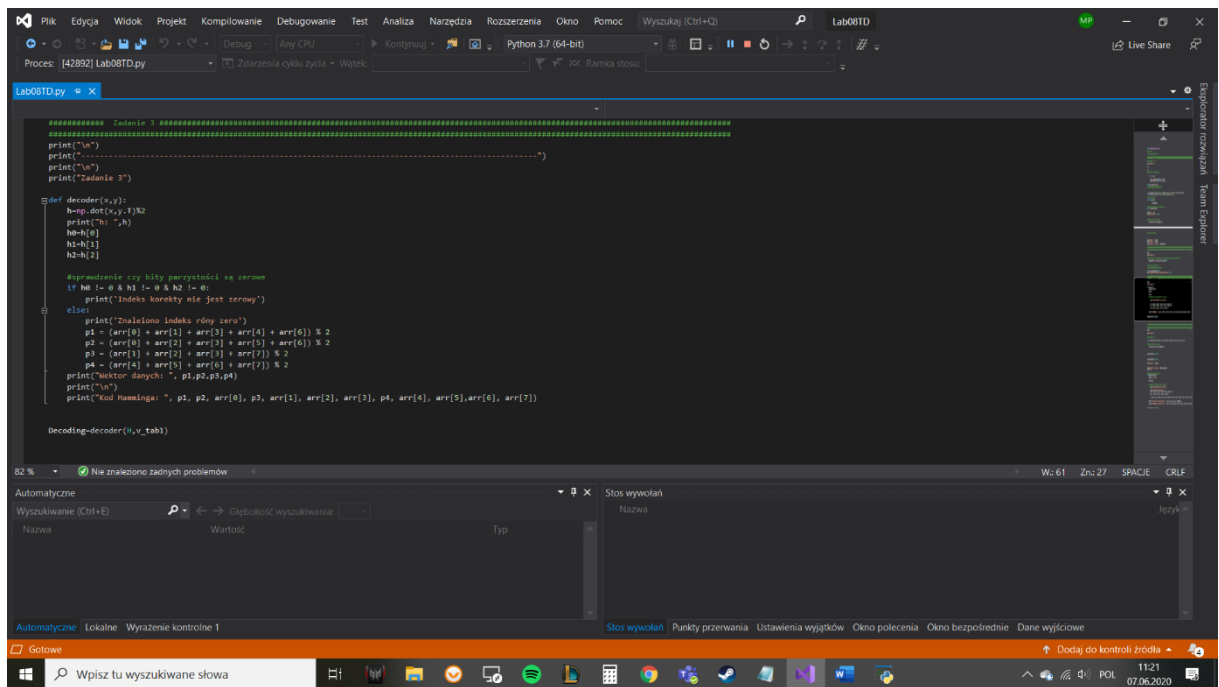
Zadanie 4
Wektor 1: [0 0 1 0 1 1 0 1]
Wektor 2: [0 0 0 1 1 1 1 0]

Wektor 1 i 2 łącznie: [0 0 1 0 1 1 0 1] [0 0 0 1 1 1 1 0]

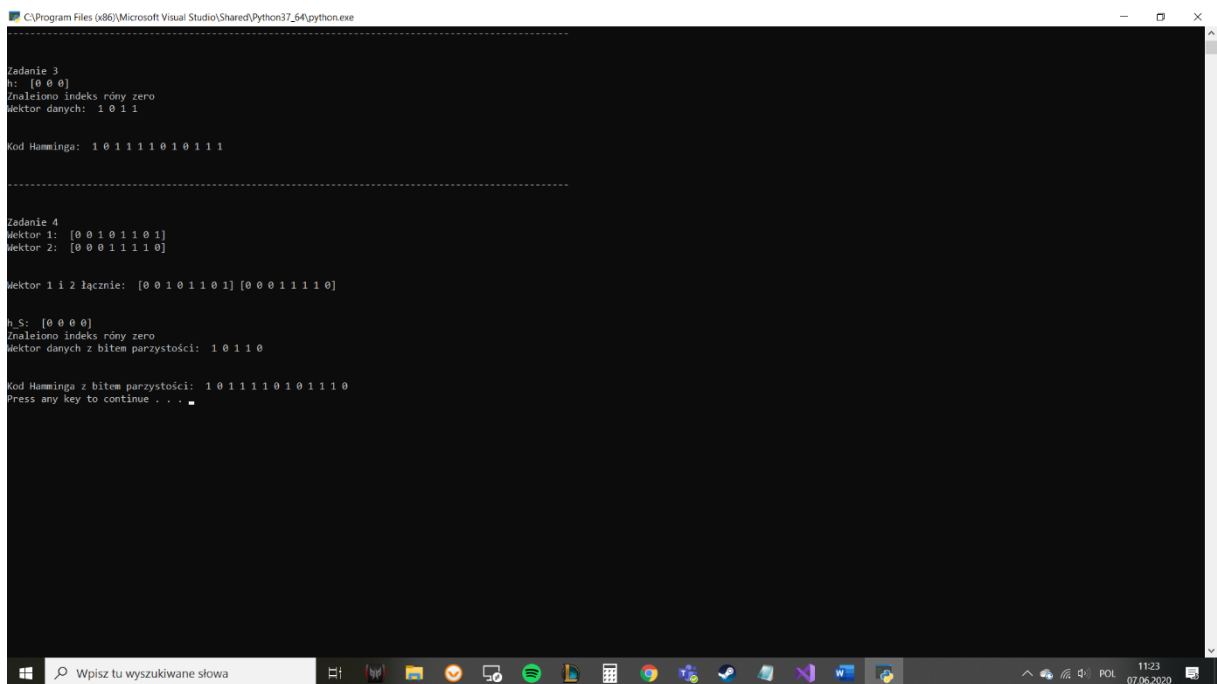
h_5: [0 0 0 0]
Znaleziono indeks róny zero
Wektor danych z bitem parzystości: 1 0 1 1 0
Kod Hamminga z bitem parzystości: 1 0 1 1 1 1 0 1 0 1 1 1 0
Press any key to continue . . .
```

Zanegowany strumień binarny z zadania 1 w zadaniu 2

3/.

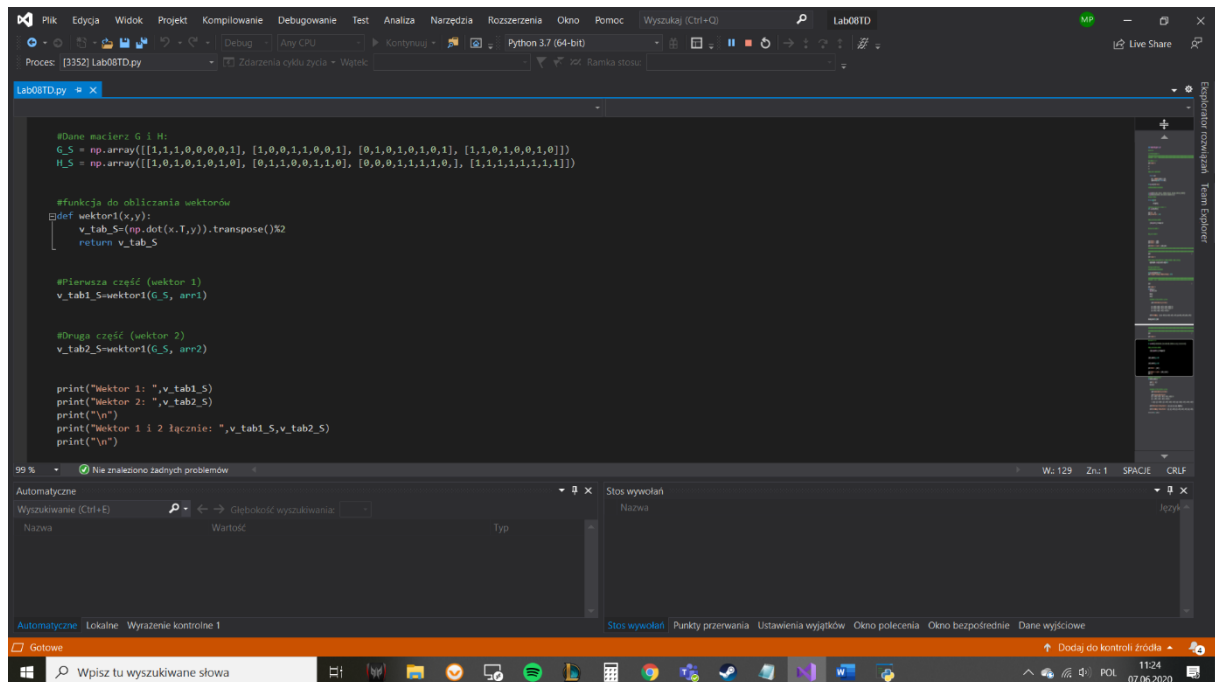


Funkcja dekodująca kod Hamminga.



Wypis z konsoli w zadaniu 3.

4/.



The screenshot shows a Python IDE with a file named 'Lab08TD.py'. The code defines two 6x5 binary matrices, G and H, and a function 'wektor1(x,y)' that calculates the dot product of a row from matrix G and a column from matrix H, returning the result as a 1x5 vector. The script then calculates two vectors, v_tab1_S and v_tab2_S, by applying the 'wektor1' function to the first and second rows of G, respectively. Finally, it prints the two vectors and concatenates them into a single 1x10 vector.

```
#Dane macierze G i H:
G_S = np.array([[1,1,1,0,0,0,1], [1,0,0,1,1,0,0,1], [0,1,0,1,0,1,0,1], [1,1,0,1,0,0,1,0]])
H_S = np.array([[1,0,1,0,1,0,1,0], [0,1,1,0,0,1,1,0], [0,0,0,1,1,1,1,0], [1,1,1,1,1,1,1,1]])

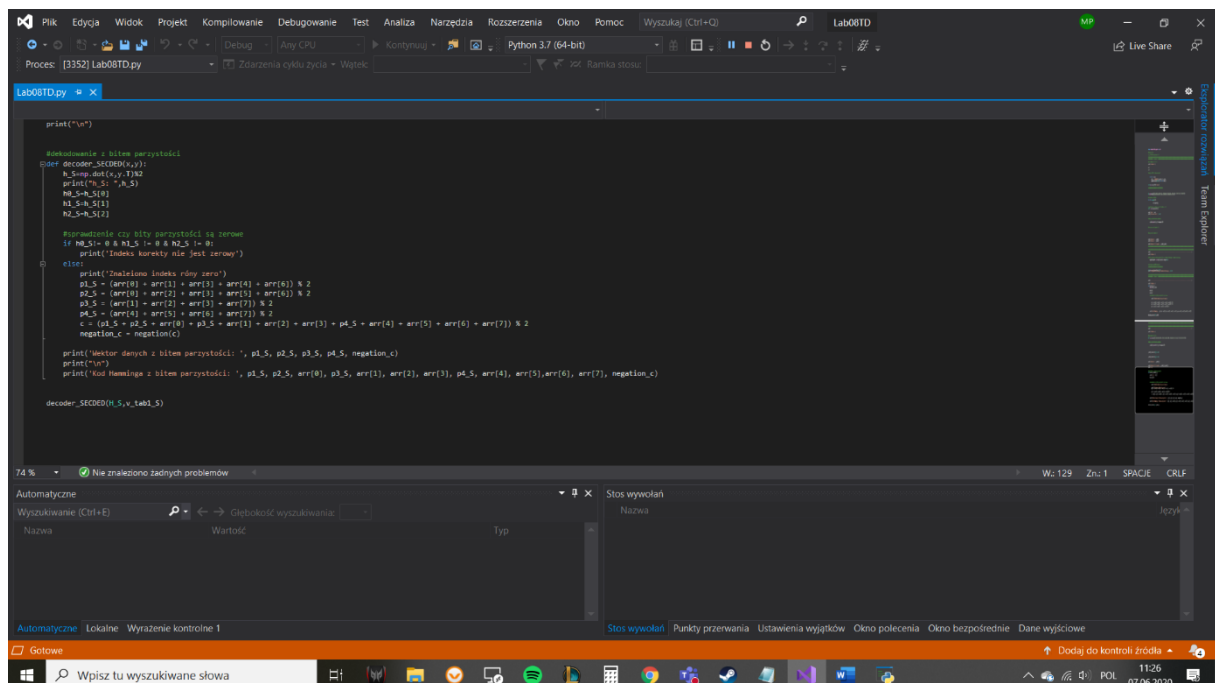
#funkcja do obliczania wektorów
def wektor1(x,y):
    v_tab_S=(np.dot(x,T,y)).transpose()#2
    return v_tab_S

#Pierwsza część (wektor 1)
v_tab1_S=wektor1(G_S, arr1)

#Druga część (wektor 2)
v_tab2_S=wektor1(G_S, arr2)

print("Wektor 1: ",v_tab1_S)
print("Wektor 2: ",v_tab2_S)
print("\n")
print("Wektor 1 i 2 łącznie: ",v_tab1_S,v_tab2_S)
print("\n")
```

Podaje dwie macierze tworze z nich wektory funkcją wektor



The screenshot shows a Python IDE with a file named 'Lab08TD.py'. The code implements a decoding process for a message encoded using a Hamming code. It starts by printing a newline. Then, it defines a function 'decoder_SECODED(x,y)' that takes a 1x10 vector and a 10x5 matrix as input. The function calculates the dot product of the vector and the matrix, resulting in a 1x5 vector. It then checks if the first bit of the vector is 0. If it is, it prints a message indicating that the index is not zero. Otherwise, it prints a message indicating that the index is zero. It then calculates the Hamming code for the vector and prints the result. Finally, it prints the decoded message.

```
print("\n")

#dekodowanie z bitów parzystości
def decoder_SECODED(x,y):
    h=np.dot(x,y)
    print(h_S,"h_S")
    h2_S=h_S[0]
    h2_S=h_S[1]
    h2_S=h_S[2]

#sprawdzanie czy bity parzystości są zerem
if h2_S[0] == 0 & h2_S[1] == 0 & h2_S[2] == 0:
    print("Indeks korekty nie jest zerowy")
else:
    print("Znaleziono indeks równy zero")
    p1_S = (arr[0] + arr[1] + arr[3] + arr[4] + arr[6]) % 2
    p2_S = (arr[0] + arr[2] + arr[3] + arr[5] + arr[6]) % 2
    p3_S = (arr[1] + arr[2] + arr[3] + arr[7]) % 2
    p4_S = (arr[4] + arr[5] + arr[6] + arr[7]) % 2
    c = (p1_S + p2_S + arr[0] + p3_S + arr[1] + arr[2] + p4_S + arr[4] + arr[5] + arr[6] + arr[7]) % 2
    negation_c = negation(c)

    print("Wektor danych z bitów parzystości: ", p1_S, p2_S, p3_S, p4_S, negation_c)
    print("\n")
    print("Kod Hamminga z bitów parzystości: ", p1_S, p2_S, arr[0], p3_S, arr[1], arr[2], arr[3], p4_S, arr[4], arr[5], arr[6], arr[7], negation_c)

decoder_SECODED(h_S,v_tab1_S)
```

Kod SECODED

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python37_64\python.exe

Zadanie 4
wektor 1: [0 0 1 0 1 1 0 1]
wektor 2: [0 0 0 1 1 1 1 0]

wektor 1 i 2 łącznie: [0 0 1 0 1 1 0 1] [0 0 0 1 1 1 1 0]

h_5: [0 0 0 0]
znaleziono indeks różny zero
wektor danych z bitem parzystości: 1 0 1 1 0

Kod Hamminga z bitem parzystości: 1 0 1 1 1 1 0 1 0 1 1 1 0
Press any key to continue . . .
```

Wypis z konsoli w zadaniu 4