In[1]:=

Karolina Tatarczyk

Kryptosystem afiniczny

In[554]:=
```
ClearAll["Global`*"]
```

In[19]:=

Część 1

In[1]:=
```
ReversibleMatrix[matrix_] := Module[{A = matrix},

If[Dimensions[A][[1]] == Dimensions[A][[2]],
det = Det[A, Modulus → 26];
If[GCD[det, 26] == 1, Return[1], Return[0]],
Return[0]
]]
Print["0 - macierz nie jest kluczem, 1- macierz jest kluczem "]
```

```
0 - macierz nie jest kluczem, 1- macierz jest kluczem
```

In[5]:=
```
A = {{7, 0}, {2, 1}};
ReversibleMatrix[A]
```

Out[6]= 1

In[7]:=
```
A = {{7}, {2}};
ReversibleMatrix[A]
```

Out[8]= 0

In[9]:=
```
A = {{2}};
ReversibleMatrix[A]
```

Out[10]=

```
0
```

In[11]:=
```
A = {{1, 1, 5}, {0, 1, 1}, {4, 7, 2}};
ReversibleMatrix[A]
```

Out[12]=
```
1
```

Część 2

In[13]:=
```
LetterForNumber[v_] := Module[{vector = v},
alphabet = {"a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l",
    "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"};
newvector = {};
vector = StringSplit[vector, ""];
For[R = 1, R ≤ Length[vector], R++,
For[S = 1, S ≤ Length[alphabet], S++,
If[vector[[R]] == alphabet[[S]], AppendTo[newvector, S - 1]];
];
];
Return[newvector]
]
vector = "matematyka";
numbers = LetterForNumber[vector]
```

Out[15]=
```
{12, 0, 19, 4, 12, 0, 19, 24, 10, 0}
```

Część 3

In[16]:=
```
NumberForLetter[V_] := Module[{vector = V},
alphabet = {"a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l",
    "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"};
newvector = {};
For[R = 1, R ≤ Length[vector], R++,
Q = vector[[R]] + 1;
AppendTo[newvector, alphabet[[Q]]];
Return[StringJoin[newvector]]
]
NumberForLetter[numbers]
```

Out[17]=
```
matematyka
```

Część 4

In[18]:=
```
CheckDimensions[matrix_, vector_] := Module[{A = matrix, B = vector},
If[Dimensions[B][[1]] == Dimensions[A][[1]], Return[1], Return[0]]
]
Code[matrix_, add_, M_] := Module[{vector = M, A = matrix, B = add},
If[ReversibleMatrix[A] == 1,
If[CheckDimensions[A, B] == 1,
vector = StringSplit[vector, ""];
For[R = 1, R ≤ Dimensions[A][[1]], R++,
If[Mod[Length[vector], Dimensions[A][[1]] ≠ 0, AppendTo[vector, "x"]
]
];
vector = StringJoin[vector];
numberMessage = LetterForNumber[vector];
dividedMessage = ArrayReshape[numberMessage,
       {StringLength[vector] / Dimensions[A][[1]], Dimensions[A][[1]]}];
encodedMatrix = {};
For[i = 1, i ≤ Length[dividedMessage], i++,
AppendTo[encodedMatrix, Mod[A.dividedMessage[[i]] + B, 26]];
];

Return[NumberForLetter[Flatten[encodedMatrix]]]
, Return["Wymiary podanych wektorów się nie zgadzają"]]
,
Return["Macierz nie jest kluczem kryptosystemu."]
]
]
```

```
A1 = {{5}};
B1 = {2};
message = "matematyka";
coded1 = Code[A1, B1, message]
```

Out[23]=
```
kctwkctsac
```

In[28]:=
```
A2 = {{5, 3}, {2, 1}};
B2 = {2, 22};
coded2 = Code[A2, B2, message]
```

Out[30]=
```
kufmkungaq
```

In[71]:=

In[31]:=
```
A3 = {{5, 2, 1}, {2, 13, 5}, {1, 7, 3}};
B3 = {2, 22, 1};
coded3 = Code[A3, B3, message]
```

Out[33]=
```
dlsuelzgktux
```

In[233]:=

In[231]:=

Część 5

In[24]:=
```
Decode[matrix_, add_, M_] := Module[{vector = M, A = matrix, B = add},
CODE = LetterForNumber[vector];
dividedMessage =
   ArrayReshape[CODE, {StringLength[vector] / Dimensions[A][[1]], Dimensions[A][[1]]}];
inverseA = Inverse[A, Modulus → 26];
decodedMessage = {};
For[i = 1, i ≤ Length[dividedMessage], i++,
AppendTo[decodedMessage, Mod[inverseA.dividedMessage[[i]] - inverseA.B, 26]];
];
Return[NumberForLetter[Flatten[decodedMessage]]]
]
```

In[25]:=
```
Decode[A1, B1, coded1]
```

Out[25]=
```
matematyka
```

Out[26]=
```
MATEMATYKA
```

In[34]:=
```
Decode[A2, B2, coded2]
```

Out[34]=
```
matematyka
```

In[35]:=
```
Decode[A3, B3, coded3]
```
Out[35]=
```
matematykaxx
```

Część 6

In[36]:=
```
frequencyattack[mes_, f_, s_] := Module[{message = mes, first = f, second = s, a, b},
firstNumber = LetterForNumber[first];
secondNumber = LetterForNumber[second];
messageNumber = LetterForNumber[message];
predictedFirstNumber = Commonest[messageNumber];
predictedSecondNumber = Commonest[messageNumber, 2];
predictedSecondNumber = If[predictedSecondNumber[[1]] == predictedFirstNumber[[1]],
     {predictedSecondNumber[[2]]}, {predictedSecondNumber[[1]]}];

solution = Solve[a * firstNumber + b == predictedFirstNumber[[1]] &&
      a * secondNumber + b == predictedSecondNumber[[1]], {a, b}, Modulus → 26 ];
A = {a /. solution};
B = b /. solution;
Return[Decode[A, B, message]]
];
message =
  "id zdatycp lxpsdbdhr zidgp bdocp qto egdqatbj oa ID ZDATYCP LXPSDBDHR ZIDGP
    BDOCP QTO EGDQATBJ OAPBPR JONLPYPR PCPAXON ROTHIDIAXLDHRX";
NumberForLetter[Commonest[LetterForNumber[message]]]
Characters[NumberForLetter[Commonest[LetterForNumber[message], 2]]]
codedMessage = Code[{{5}}, {3}, message];
frequencyattack[codedMessage, "e", "t"]
```
Out[38]=
```
d
```
Out[39]=
```
{d, p}
```
Out[41]=
```
stztevuobhpbqtjtnlzstibjtwobgvwyitgevjxweeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee⋱
    eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
```