Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Фізико-технічний інститут

ЛАБОРАТОРНА РОБОТА №2

з дисципліни

«Криптографія»

на тему: «Криптоаналіз шифру Віженера»

Виконали:

студенти 3 курсу ФТІ

групи ФБ-73

Маковецький Андрій та Бадарак Оксана

Перевірили:

Чорний О.

Савчук М. М.

Завадська Л. О

Мета роботи:

Засвоєння понять ентропії на символ джерела та його надлишковості, вивчення та порівняння різних моделей джерела відкритого тексту для наближеного визначення ентропії, набуття практичних навичок щодо оцінки ентропії на символ джерела.

Порядок виконання роботи:

0. Уважно прочитати методичні вказівки до виконання комп'ютерного практикуму.

1. Самостійно підібрати текст для шифрування (2-3 кб) та ключі довжини r = 2, 3, 4, 5, а також довжини 10-20 знаків. Зашифрувати обраний відкритий текст шифром Віженера з цими ключами.

2. Підрахувати індекси відповідності для відкритого тексту та всіх одержаних шифртекстів і порівняти їх значення.

3. Використовуючи наведені теоретичні відомості, розшифрувати наданий шифртекст (згідно свого номеру варіанта).

Ключі для зашифрування:

2: 'ор',

3: 'рик',

4: 'кусь',

5: 'морти',

9: 'автопилот',

12: 'велоцераптор',

15: 'астроориентация'

Індекси відповідності зашифрованого тексту:

Довжина ключа	Індекс відповідності
0	0.053695
2	0.044587
3	0.039373
4	0.0358
5	0.037049
9	0.034776
12	0.034023
15	0.03495
20	0.032833

Розшифрування тексту (варіант 11):

Індекси відповідності для довжин ключа 2-30

2	0.035446
3	0.035486
4	0.035423
5	0.035516
6	0.035521
7	0.035473
8	0.03546
9	0.035554
10	0.035477
11	0.035286
12	0.035625
13	0.035453
14	0.035314
15	0.035454
16	0.03557
17	0.058332
18	0.035532
19	0.035478
20	0.0352
21	0.035777
22	0.035293
23	0.035422
24	0.035629
25	0.03532
26	0.03525
27	0.035399
28	0.035334
29	0.035563
30	0.035619

При r = 17 індекс відповідності шифротексту значно більший за інші,
отже довжина ключа — 17 символів.

Після знаходження довжини ключа виконуємо розшифрування шифру Цезаря для кожного блоку за допомогою частотного аналізу. При порівнянні найчастіших літер у блоках та найвірогідніших літер мови, отримуємо такий ключ: венецианскийкужец
Ключ скоріше за все складається з двох слів, перше з яких — *венецианский,* тому треба перебрати друге слово. Повторивши розшифрування Цезаря для 14го блоку ще 2 рази отримуємо слово *купец*.
Отже, вірогідніше за все, ключ: венецианскийкупец.
Розшифрувавши шифротекст цим ключем, отримано змісовний текст, отже ключ підтвердився.

КЛЮЧ: венецианскийкупец

Шифротекст:

втяугроъцсхйиббыеумчцтптикуочяькуфупчхлоюгжкйцтарсъшяуьнныфонингвциюфыовилъсвнфтюлйдг
ашьицсывьилхтфчнфуэуъртцяцыпюраэпеябчнсюэещфпаьехехацидмырмрцшсжчдуещущцсттйырчуббв
пкяхймнывкуйъыьушэйаъдфмтипъоыпюудмкнтйлдтукасмшънневзикзыдныкткшцпчыкнпкбдмычткчоыъб
еээъехчрызпщъттыужупндзчртшънцжшыцврчэдихаяяълчмйфзвзрчнлятыыхицйсбцхпнфпдрмюашяыпал
квмурйццнхьпъиьапчавтиъашышнйэъкюптюрфызышьяцпщфтфочцмххцацвньщцаьысцыщшпцикаомхрк
ьуысдкцщуысннхпоншьожссуючдзнъяышдмуъчжвзаьицбфюкъешещшъвзтчышиюыкуцкэпхивърешинх
щлыюьоъгчроьхыммтгбъчцбтжспкайцяущюпчщпчскпвчйсыхяомчнъшяькгпупижысянщцлпгтебуешежр
нывьынйяэозхфсалинйццзлхыдужвйчкчгдэярифшеыазнндчдфоуцькхшгфшжвинтгидтькъечшыущгапнън
тйрбиъшхюкзрьъалхепвщцхчысэюрстрхэиыбтьйявякъучнзюубиышшйлюлзезцчкэивмшврхнюпзйупшугр
вещцхсршжквгученъоозпучмуббздулсдлишдмюоъэснзоуяхххачсцхссчптюбцпдицгыыктхшцрахпкпцецм
ъщьдъфуъуевцъалятыжъышфышсдлпыхцйлйцокйьбъпгхзпцычрмюшщытгпцзэфнрюйыпушмьтхэргэуор
ытлхтмфчтлфравтацбцвыэбъчцбфждееяцикоюгкуччыжквксыибрбмялеышяушввчйтымущсйчщтеэснфут
цбрбясфщфэкчрдубщтычрхйхцъжфкмцехациртйюплчмбянизмъефзаъгшхсшцяшзфнячжнвычкщесуаздкч
ызцшынюьцтбькидкэбинмъцлуйнбуежацайтйущяушсыэькджтысйзвпцърфыжутйпкыйгцмащцнъьжауз
фумттнмыцчнхпгччзбчтпйбищфшмчцтькщтшжшюпзнэшрюьбсежрзюебирхюшъчнчпзсйтньюъвшплуочо
птиртхуеысяяпщйхуянгрттзбжбшцчгыкэапцикщзсчедсхдцеъпчыоьяушгнтупщохоччднбчувцгшщлщхптб
бзбзичшнрсрйкоышъмцфкщьицнтфъывэчсшбкьъаязнавфуичжабиржыожцдхгщшсъбуезфхнтггхшпонтш
чьнщнефкфъивяяцаэещеасуьшщийавхгбкхзнядушагтусбэлспщфтцднспцтучвэщутдъаивпдчдкушмлтосж
рагзфыпцоуяыхзцтдлццоттцицрдгшпйлустцъшяпцкхыхйъккдаегкушужннгятлщкйчегрцнрцхиыушъкхут
ужрйъаяшосщбкйвфпцзвтхущшагщкхчтюэхыпыыцгрмьбшбйуефссдраьонмытгнъьхфузфепнвкаювкуйъы
ьудучнрззбмиьцкмуахцйзтцыуиянчцлшеозюишяклттыукфншгэлывтяугропэшрюнюпмцчыттцкащхшшчн
уайцзчюдвхедгшкйычфрцйупширрнхекдщгфйбриашъилхгжщиыуъежктыфжрвтгмихнбафджеоезщаъщш
щсчпэхспучущмауэеччыяфквудчущмапмбъчьбачцннъкбждхэещйхуянгрйыцйлвнйцгнпюччтеуяушгспсц
ръкюпхюпухццуаъщшыюшчочбрхттмцкосыщйчыэцюпбыхжизпмкхышачугвэшнвгвнвшщыкхчсгрфэуоы
кытпмьчшюуэвжичтлдтэемчхщьазроянздбнвыицтбюхюжшъешнръяншйаыптдунъбдшаъгшхсшхчййдею
фбцхыъщнапефцтурхэацмпмшйфкцпъвкхнпицивгъыншяжхыйстхггмьяфышкшбчытдтчюэкнытпхпачрю
убпацхтыютцицыяцнкчнгннмыюпщыжцяемкеъувррюзпхйнчфшшудчущцеюгжшчхпыухехацихарбкюска
эсгзлсеяъеэяхдбэепфйупнодъсщцяикэчвзыубпсдшщхюкэшэдбббхьекенчтюжцымыьещрххчннмюгехоьд
фхъшкычизжтьеэлсчэъддмньыфжтипучмшщшзфкьрдскэямдзыыукиюфыйдйныэъихшгьувхфэкътуюакъе
чуозйкрхъшцрнгжоохевъдлуяхцпдэсрнцжтарйъецпциняячрчьшрдбажшгхлопяърымбытынгоушдеюгжуз
оывпдфуэалуигсщцуъобаенкъпдстыичцмхуубчррычццнхжъицйеьежрьъугнпыхмрпчбачтщчыждйщнрцц
фмучсетньнзилнвпшепъьузфбщшъшоюгжрмхжруакоасющлыучцмшхэххфтнсхцрныэуушцуешзюнгеыся
нтчоыафрцзчысдсаъгшхсшъьефбчнпюэчяяцынъоынзнапшиуиенщцышявьиртхъылоъцнцлшгочирисеаик
фснцлшгчздпнякжпашщлыбтефсафухъзуыеслусрачъъххнпцфиъсскйхфкзыттыйццбкгшфшшдъкгрттрдки
ямчишъыыегмшрхйщтхгктьидъешхлнраыюэмлнбфжюуяжкщрдшеъзшхьщбщеетужеяипэящцлпчлдартд
шецооцоилхбякгшухчцтвнвшщыкхъдыойыучнднаърнпеадвпкоаидмахъняшябеаксокэфошучгхнбсужкчй
тымаюгншйаыптдехныюиныхкччрыснзрсъуфлоссокйсхвпщррыццъыюушнмшъпжйжкебцхтыютццэчъди
зжмдзъаъдлцфьжьувехныюиныяьусбаэзыжбщубяаънпэчъбзушмуьыхыврсгукиуешщнючсэдтукэмъуенц
пухрдшеъзшыюшчочцпчытюцгсцыфдыюскщрцшушихосыщйчыэижвыхегцгыушшсьффцарттьгцмъянцш
эдбдарзоубдштипфуьбънчрзпкгнцхщплчъуацйийттюзэяюкйсшятцсэттююыюаъзыкаьаыдйпшлфеэсяфиф
ыьанфуоаюэннъърцкчэнзмеябшнсйхпхекапоъзэйшшрдхйжяышычцавчйтыщтыщхцлпафыюбшшнмиввяо
рыхуъыынуярцхчтьшъушафьгрцызыщтйэшзшшъсубкчщтыщбтчкъешемчдеуунъимыцнцюшъонгвжтцвнн
мгктлшеччднпнкиъачушъстдщшовкяидкоэщьлчлфэрцпьтрдъгытлцншфяаянеъыыуоящхрншфяаяеуождрл
ххшйщеьъегщкшуилоотшчыыечтденпъдмбтфткчмдфчхипхкймиэсуцыысуецуупкзъьрямцщтеькисуючдсч
твьхдуюптнзрычецсяяуяоыеьаяеуождрлхыктлелфцавмнтдяеюгчнтвышрцпьтрдъгытлццпжунжвояуехиъя
нцтчумчюоаюрюуасюшиюъурхслййдшцлпцхрыцафцанесашитйашосьэчзехчйидкоэщьйоыяпхоеупужрто
цышйоырущвцыжышиюнымьябыиддуэнийеющхыштюпйгреюушнсянццимшзеыфцмтцаелыццоцжакжж

ыанвыдэянцлшгччвродкзъниъошьнюптнзрычшндйгещдчкфципурудъцнщхръфбякыуаъьыщтъяуфйьшян
ерчысйятывфиркелфжвзсшдъеггшфчуафцаррйпдтачтееышхкхцйнябззояхккйхкфсиржирйхерязъйфышф
жкзчшзуасюшщчмшачтоттидкоэщьуйчкфрдфттэыкешщыдшшлфзыннпеящярямцщтеркзпнюсыщтнфшкч
ъыбцддкючтщопцыъенбсужафэешрлйюшъдыбскихкебыщлхашксчбсеиюцмцдкеюгтхйобытырцяеидсмдр
рнкяэкщръымхрннсшхышвяузфнщкгзгывщцнтдпсштускъдяпяхийбеэжсхйэеидоячтмйщгчйыфцмфдкиъя
миыждорймепувыапцодччеэцшвэтидчофушуочыоныучйццфдйрмцфтеэфжвзсенъоущокчщюэюыптиоъдя
пяхршшдэзыучхидкоэрыцлпдббврдукзъочяынщоапдзрзтцидеюьтццкяькзрзтчйнтывьиыждошькйнжыщм
ъцоиоьфэрызйэшьнсчщущчмшбдивпхшънрахжзлшюыуюсдяпюттпятфьыювицошскжыввяорыыепхслиы
жчцсчяъпсчээощржоувцлыхшсъталужупнлюъеьярифэьачцмеччйъпэяуъсфдчттрхалюмушсчяохббззфуэу
гыоъцлцнкрбувротйюхоэяохдуббкмртюрычтныыучмаэквхттчдятыапщушяппжхфъавлрнутнхнюпмцйею
аеилыюпырчуфчвзтмцслрнпыхсцэппйатхжймьегэтьнбрждсудчыхнтъвртцбъуъроюнгдоэвонфкьбквкрыц
идкныцхощгэоикпюнгдоэдррнэюптнзрычцчопцъхсьшьеетепуешиьцъкчцвэздтуякгцпэщыихшяюкждущф
дкоэяьншшхфхгкчттцуепяоиъцббхюфкьхюхаюгшзвпябуерзыхдъеъцщцлшгпьюдецчыхиъсщшймбьнгврт
здивыбшяйуеффжбстъдхьчяфмчцжхнъвиыжчцспрьгоэцэйгкпхчжыдъялынпеюуевцябгиннвьцигжнйдтур
шшгнтэшооышапцйеябвхеъцфртхуьыныуяьицуъцнцптхфчькзтчйхерятусчшбрцпьтрдъвифшкчъыбкдоушу
ъцмющчдчъшъегщкшуинвюгшшччжсуисуруърттыктьитакящеюнзъоэщишйсхфййучнхютупмнцлшгччыр
сырдмощвтхешгфрцпьтрдъерялйчфущуяфещцюхъыбмибкячрдучйцсхбтхчцмшкфрддкчедъегцнцюшелви
рдюевлопяъжапдбтслтыунннйтпмцеьжкбрзтплцтмтхимшчпххгуреэмоэямгтхуьыынуятйттытеъйцачжснк
рудъегэелчмбянврсырдмощвтхешгфрцпьтрдъерялйчшъьеарсысшиштъцрфшдбнетуючдуаъипучьшашъвх
йрцхчтьшъцшвяуохзнцзъаэщфъчлызбйоушдфкаювхнныескгпупащцвыбтъошяэрджущцюхъыбпуйчкфрд
фкручйоыькхапюэчыуфымшцлтютзлклфовкрыцирлнюбнфшеарыжцязыхныяырцжбякбвтдкбцттюплчмб
янизуюнгдоэцхицбшъуизжнчфакнэшсслбмчдфсырдмоцвтхешгаущнеютдкошцыынпфчхдмехзззкхжьиут
ивьыпавзыбецуъцнпеяклукючышпнбштсщгэьючуяхцщлтъчиьфыукяучпнинлйюшщушажебудхрюнсшщц
цпчбсажвмхчтщяьбшбошеээынзшшнаицэтшмшфрущрцьуъруьсырнкхфйтоешбныгнюлгфтдбнгпащфдщо
вяцфчпьачцтуючюрсящчхчуяэкясусфдшоцькхапюэчгзшучдояьяяцпчуюебмшхрюхаосхтьчуюузръхрюхаь
яоччъэвзсокъдгнуфюкныпфпчфъпнидйбыяяхоьтсезпкфтцжмыьшмчудчттрхъуешоедмиъыэплюфкщтычр
нуоыаьбыуяоешбнркаашчуцэцерййкщцдайэосмыънерстхбиндцхцычшвлирзитыызъспъоцшьдчвчлэаигы
тлцчяцэхыбзтйтчодгтяышбарысттфжрзъсикйыюптнзрычгыпыаъилошуеьзжайтывнвнуйсусфдтдспыкыхг
шфчнючжспкамгитйэпхиэяфтирчычыючяяэпцкшбцдгцязыхныфшшолшьпцнестдщтнбттимуызззхнцзтау
дщшчмузкщрцитцщтнюшксьдотцмушкгрбшснцъхбснвзтмфживссоцфрапзслчхтцщвтгэйсудбзцжушидщк
эммиыжафртйдччдяецвехжьбапжэчйсдоныюшкушаекартгушчрнуоилеьукипеэшьы

Розшифрований текст:

антонионезнаюотчегоятакпечаленмнеэтовтягостьвамяслышутоженогдеягрустьпоймалнашелильдобылчт
осоставляетчтородитеехотелбызнатьбессмысленнаягрустьмоявиноючтосамогосебяузнатьмнетрудносала
риновыдухоммечетесьпоокеанугдевашивеличавыесудакакбогатеиивельможиводильпышнаяпроцессиямо
рскаяспрезреньемсмотрятнаторговцевмелкихчтокланяютсянизкоимспочтеньемкогдаонилетятнатканыхк
рыльяхсаланиоповерьтееслибятакрисковалпочтивсечувствабылибтаммоисмоейнадеждойябыпостояннос
рывалтравучтобзнатьоткудаветерискалнакартахгаваниибухтылюбойпредметчтомогбынеудачумнепредве
щатьменябынесомненновгрустьповергалсалариностудямойсупдыханьемявлихорадкебыдрожалотмыслич
томожетвмореураганнаделатьнемогбывидетьячасовпесочныхневспомнившиомеляхиорифахпредставилб
ыкорабльвпескезавязшимглавусклонившимнижечембокачтобцеловатьсвоюмогилувцерквисмотрянакамн
изданиясвятогокакмогбыяневспомнитьскалопасныхчтохрупкиймойкорабльедватолкнуввсепряностирасс
ыпалибывводуиволныоблеклибвмоишелканусловомчтомоебогатствосталоничемимоглибяобэтомдумать
недумаяпритомчтоеслибтакслучилосьмнепришлосьбызагруститьнеговоритезнаюянтониогруститтрево
жасьзасвоитоварыантонионетверьтемнеблагодарюсудьбумойрискнеодномуявверилсуднунеодномуимест
усостояньемоенемеритсятекущимгодомянегрущуиззамоихтоваровсаларинотогдавызначитвлюбленыанто
ниопустоесалариноневлюбленытакскажемвыпечальнызатемчтовыневеселыитолькомоглибсмеятьсявытв

ердяявеселзатемчтонегрущудвуличныйянусклянусьтобойродитприродастранныхлюдейоднiглазеютихох
очуткакпопугайуслышавшийволынкудругиеженавидкакуксускислытакчтовулыбкезубынепокажуткляни
сьсамнесторчтозабавнашуткавходятбассаниолоренцоиграцианосаланиовотблагородныйродичвашбассан
иограцианоилоренцоснимпрощайтемывлучшемобществеоставимвассаларинооталсябячтобвасразвесели
тьновотявижутехктовамдорожеантониовмоихглазахценавамдорогасдаетсямнечтовасделазовутирадывып
редлогуудалитьсясалариноприветвамгосподабассаниосиньорынокогдажмыпосмеемсякогдавычтотостал
инелюдимысаларинодосугвашмыделитьготовысвамисаларинoисаланиоуходятлоренцокбассаниосиньорр
азвыантонионашлимывасоставимнопрошукобедунепозабытьгдемыдолжнысойтисьбассаниопридунаверн
оградианосиньорантониовидувасплохойпечетесьслишкомвыоблагахмирактоихтрудомчрезмернымпокуп
аеттеряетихкакизменилисьвыантониоямирсчитаючемонестьграцианомирсценагдеувсякогоестьрольмояг
рустнаграцианомнеждайтерольшутапускайотсмехабудувесьвморщинахпустьлучшепеченьотвинагоритче
мстынетсердцеоттяжелыхвздоховзачемжечеловекустеплойкровьюсидетьподобномраморномупредкуспа
тьнаявуилихворатьжелтухойотраздраженьяслушайкаантониотебялюблюяговоритвомнелюбовьестьлюди
укоторыхлицапокрытыпленкойточногладьболотаонихранятнарочнонеподвижностьчтобобщаямолваимп
риписаласерьезностьмудростьиглубокийумисловноговорятнамяоракулкогдавещаюпустьипеснелаетомой
антониознаюятакихчтомудрымислывутлишьпотомучтоничегонеговоряттогдакакзаговоривонитерзалибу
шитемктоихслышаближнихдуракаминазвалбывернодаобэтомпосленонеловитынаприманкугруститкакуюс
лавужалкуюрыбешкупойдемлоренцонупокапрощайапроповедьякончупообедавлоренцоитаквасоставляе
мдообедапридетсямнебытьмудрецомтакимбезмолвнымговоритьнедастграцианограцианодапоживисомно
югодадвазвукголосатысвоегозабудешьантонионудлятебяястануболтуномграцианоотличноведьмолчанье
хорошовкопченыхязыкахдавчистыхдевахграцианоилоренцоуходятантониогдесмыслвегословахбассанио
грацианоговоритбесконечномногопустяковбольшечемктолибоввенециигорассужденияэтодвазернапше
ницыспрятанныевдвухмерахмякинычтобыихнайтинадоискатьвесьденьанайдешьувидишьчтоискатьнест
оиловенецияулицавходитланчелотланчелотконечносовестьмояпозволитмнесбежатьотэтогожидамоегохо
зяинабесменятаквотитолкаеттаквотиискушаетговоритгобболанчелотгоббодобрыйланчелотилидобрыйго
бboилидобрыйланчелотгоббопустиногивходбегивовсетяжкиеудирайотсюдаасовестьговоритнетпостойче
стныйланчелотпостойчестныйгоббоиликаквышесказаночестнейшийланчелотгоббонеудирайтопниногой
наэтимыслиладноахрабрыйдьяволвелитмнескладыватьпожиткивпутьговоритбесмаршговоритбесрадибог
асоберисьсдухомговоритбесилупиладноасовестьмоявешаетсянашеюкмоемусердцуимудроговоритмойче
стныйдругланчелотведьтысынчестногоотцаилискореесынчестнойматерипотомучтосказатьправдуотецто
мойнесколькокакбыэтовыразитьсяотдавалчемтобылунегоэтакийпривкусладносовестьмнеговоритланчел
отнешевелисьпошеveливайсяговоритбеснисместаговоритсовестьсовестьговорюправильнотысоветуешье
слиповиноватьсясовестинадомнеостатьсяужидамоегохозяинааонтопростименягосподисамвродедьяволаа
чтобыудратьотжидапридетсяповиноватьсялукавомуаведьонтосвашегопозволенияиестьсамдьяволитопра
вдачтожидвоплощенныйдьяволипосовестиговорясовестьмояжестокосерднаясовестьеслионамнесоветует
остатьсяужидабесмнедаетболеедружескийсоветятакиудерудьяволмоипяткиктвоимуслугамудерувходитс
тарыйгоббоскорзинкойгоббомолодойсиньорскажитепожалуйстакактутпройтиксиньоружидуланчелотвст
оронуонебодаэтомойединородныйотецонслептакословноемунеточтопескомакрупнымгравиемглазазасыпа
лонеузнаетменясыграюснимкакуюнибудьштукугоббопочтеннейшиймолодойсиньорсделайтемилостькак
мнепройтиксиньоружидуланчелотаповернитенаправоприпервомповоротеноприсамомпервомповоротепо
вернитеналеводасмотритепринастоящемтоповоротенеповорачивайтенинаправониналевоаворочайтепрям
ехонькокдомужидагоббосвятыеугодникитруднобудетпопастьнанастоящуюдорогувынеможетесказатьмн
енекийланчелотчтоунегоживетживетунегоилинетланчелотвыговоритеомолодомсиньореланчелотевсторо
нувотпогодитекакуюясейчасисториюразведустарикувыговоритеомолодомсиньореланчелотегоббокакойт
амсиньорвашамилостьсынбедногочеловекаотецегохотьэтоясамговорючестныйнооченьбедныйчеловекхо
тяблагодарябогаздоровыйланчелотнуктобытамнибылегоотецмыговоримомолодомсиньореланчелотегобб
оознакомомвашеймилостипростоланчелотесударьланчелотнопрошувасстариктобишьумоляювасследстве
нновыговоритеомолодомсиньореланчелотегоббооланчелотеспозволениявашеймилостиланчелотследстве
нноосиньореланчелотенеговоритеосиньореланчелотебатюшкамойибоэтотмолодойсиньорсогласноволесу
дебирокаивсякихтакихученыхвещейвродетрехсестерпарокипрочихотраслейнаукидействительноскончал

сяилиеслиможновыразитьсяпрощеотошелвлучшиймиргоббогосподиупасидаведьмальчуганбылистинны
мпосохоммоейстаростиистинноймоейподпоройланчелотнеужтожяпохожнапалкуилинабалкунапосохили
наподпоркувыменянеузнаетебатюшкагоббоохнетяваснезнаюмолодойсиньорнопрошувасскажитемнеправ
дучтомоймальчикупокойгосподьегодушуживилипомерланчелотнеужтовынеузнаетеменябатюшкагоббоо
хгореяведьпочтичтоослепнепризнаювасланчелотнупоправдедажебудьувасглазавпорядкевыитомоглибын
еузнатьменяументототецчтоузнаетсобственногоребенкаладностарикявамвсерасскажупровашегосынаста
новитсянаколениблагословименяправдадолжнавыйтинасветубийствадолгоскрыватьнельзякточейсынэто
скрытьможноновконцеконцовправдавыйдетнаружу

Код програми:

```
import os
ALPHABET = 'абвгдежзийклмнопрстуфхцчшщъыьэюя'
ALPHABET_DICT = {
	'а': 0, 'б': 1, 'в': 2, 'г': 3, 'д': 4, 'е': 5, 'ж': 6,
	'з': 7, 'и': 8, 'й': 9, 'к': 10, 'л': 11, 'м': 12,
	'н': 13, 'о': 14, 'п': 15, 'р': 16, 'с': 17, 'т': 18,
	'у': 19, 'ф': 20, 'х': 21, 'ц': 22, 'ч': 23, 'ш': 24,
	'щ': 25, 'ъ': 26, 'ы': 27, 'ь': 28, 'э': 29, 'ю': 30, 'я': 31
}
KEYS_DICT = {
	2: 'ор',
	3: 'рик',
	4: 'кусь',
	5: 'морти',
	9: 'автопилот',
	12: 'велоцераптор',
	15: 'астроориентация',
	20: 'баллистокардиография'
}
THEORETICAL_FREQUENCIES = {'ч': 0.015034171004991752, 'у': 0.029795188208325298, 'ж':
0.011383193721390263,
'о': 0.11560475894623333, 'й': 0.010619077204333326, 'м': 0.03125914975969607, 'и':
0.06689054566488849,
'р': 0.041951425041597934, 'е': 0.08193364326470567, 'т': 0.06399297298455342, 'л':
0.049303368539823325,
'я': 0.023675114796009453, 'п': 0.026886903614199712, 'ш': 0.010020995351029415, 'в':
0.03993222929208532,
'к': 0.03429062136241261, 'с': 0.05151716405653034, 'н': 0.0644642971913362, 'а': 0.08244960044561561,
'б': 0.016871264220065556, 'ы': 0.02083824296048732, 'г': 0.016869478901100472, 'ю':
0.0064396455070662925,
'щ': 0.0035456434646613964, 'д': 0.029977290742764104, 'ъ': 0.022491448322157236, 'х':
0.008805193135805644,
'з': 0.016833772521798743, 'ц': 0.0030957430854596482, 'э': 0.0028065214131156673, 'ф':
0.00042133527576036734}

def import_data(filepath):
	with open(filepath, 'r', encoding='utf-8') as data_source:
		return data_source.read()
```

```
def vigenere_encrypt(plaintext, key, alphabet_dict):
	reverse_alphabet_dict = {val: let for let, val in alphabet_dict.items()}
	period = len(key)
	ciphertext = ''
	for s in range(len(plaintext)):
		pt_value = alphabet_dict[plaintext[s]]
		key_value = alphabet_dict[key[s % period]]
		ct_value = (pt_value + key_value) % len(alphabet_dict)
		ciphertext += reverse_alphabet_dict[ct_value]
	return ciphertext

def vigenere_decrypt(ciphertext, key, alphabet_dict):
	reverse_alphabet_dict = {val: let for let, val in alphabet_dict.items()}
	period = len(key)
	plaintext = ''
	for s in range(len(ciphertext)):
		ct_value = alphabet_dict[ciphertext[s]]
		key_value = alphabet_dict[key[s % period]]
		pt_value = (ct_value - key_value) % len(alphabet_dict)
		plaintext += reverse_alphabet_dict[pt_value]
	return plaintext

def vigenere_encrypt_lab(filepath, keys, alphabet_dict):
	plaintext = import_data(filepath)
	for key in keys:
		output_file_path = os.path.splitext(filepath)[0] + '_encrypted_keylen_' + str(key) + '.txt'
		with open(output_file_path, 'w', encoding='utf-8') as output_file:
			output_file.write(vigenere_encrypt(plaintext, keys[key], alphabet_dict))

def calculate_index_of_coinsidence(text, alphabet_dict):
	n = len(text)
	res = 0
	letters_count = {}
	for letter in text:
		if letter in letters_count:
			letters_count[letter] += 1
		else:
			letters_count[letter] = 1
	for letter in letters_count:
		res += letters_count[letter] * (letters_count[letter] - 1)
	return res / (n * (n - 1))

def get_letters_counts(text):
	letters = {}
	for i in text:
		try:
			letters[i] += 1
		except:
			letters[i] = 1
```

```
    return letters

def get_most_frequent(text):
    letters = get_letters_counts(text)
    rev = {value: key for key, value in letters.items()}
    return rev[max(rev)]

# decipher Caesar cipher for integer key
def decipher_Caesar(text, key, alphabet_dict):
    rev_AD = {num: letter for letter, num in alphabet_dict.items()}
    deciphered_text = ''
    for letter in text:
        encrypted_letter_value = alphabet_dict[letter]
        decrypted_letter_value = (encrypted_letter_value - key) % len(alphabet_dict)
        decrypted_letter = rev_AD[decrypted_letter_value]
        deciphered_text += decrypted_letter
    return deciphered_text

# break Caesar cipher using frequency analysis
# call the next iteration if the text does not match
# (specificly done for vigenere decryption)
def break_Caesar(text, theor_freq, alphabet_dict, iteration=1):
    sorted_theor = sorted(theor_freq.items(), key=lambda kv: kv[1], reverse = True)
    rev_AD = {num: letter for letter, num in alphabet_dict.items()}
    current_theor_letter = sorted_theor[iteration - 1][0]
    current_theor_value = alphabet_dict[current_theor_letter]
    most_frequent_in_text = get_most_frequent(text)
    most_frequent_in_text_value = alphabet_dict[most_frequent_in_text]
    probable_key = (most_frequent_in_text_value - current_theor_value) % len(alphabet_dict) # ?
    decrypted_text = decipher_Caesar(text, probable_key, alphabet_dict)
    return (decrypted_text, rev_AD[probable_key])


def break_Vigenere(text, alphabet_dict):
    global THEORETICAL_FREQUENCIES
    reverse_alphabet_dict = {value: key for key, value in ALPHABET_DICT.items()}
    #
    #       Find key length
    #
    IC_dict = {}
    for block_len in range(2, 31):
        ic_sum = 0
        for i in range(block_len):
            seq = ''
            for j in range(i, len(text), block_len):
                seq += text[j]
            ic_sum += calculate_index_of_coinsidence(seq, alphabet_dict)
        res = ic_sum / block_len
        IC_dict[block_len] = res
    avg = sum(IC_dict.values()) / len(IC_dict)
```

```
	print('Average IC:', avg)
	print('Possible key length variants:')
	possible_key_len_dict = {}
	max_ic = 0
	k_len = 0
	for k, v in IC_dict.items():
		print('{:>2} {:.6f}'.format(k, v))
		if v > max_ic:
			max_ic = v
			k_len = k
	print('\nKey length: ')
	for key, value in IC_dict.items():
		if value == max_ic:
			possible_key_len_dict[key] = value
			print(key)

	#
	# For every possible key length try to decipher Caesar
	#
	for key_len in possible_key_len_dict:
		print('\nTrying to decipher for key len =', key_len)
		caesar_sequences = []
		for i in range(key_len):
			caesar_sequences.append('')
		for i in range(len(text)):
			caesar_sequences[i % key_len] += text[i]
		deciphered_sequences = []
		key = ''
		for i in range(key_len):
			deciphered_sequences.append('')
		for seq_num in range(len(caesar_sequences)):
			deciphered_sequences[seq_num], key_part = break_Caesar(caesar_sequences[seq_num],
THEORETICAL_FREQUENCIES, alphabet_dict, 1)
			key += key_part
		result_text = ''
		for i in range(len(deciphered_sequences[0])):
			for seq in deciphered_sequences:
				if i < len(seq):
					result_text += seq[i]
			result_text += '|'
		#
		# Manual improvements using CLI
		#
		print('Possible key:', key)
		print('Check if the text is OK and change the letter of the key if needed\n')
		print(result_text)
		while True:
			print('\nChoose an option:')
			print('1. Retry one of the blocks')
			print('2. View current text')
```

```
                print('3. View current key')
                print('4. Finish and write the deciphered text to file')
                option = input()
                retry_iteration = 2
                if option == '1':
                        ind = int(input('Enter the index of the character to retry: '))
                        while True:
                                new_result_text = ''
                                new_key_list = [c for c in key]
                                print(new_key_list)
                                deciphered_sequences[ind], key_part1 = break_Caesar(
                                        caesar_sequences[ind], THEORETICAL_FREQUENCIES,
alphabet_dict, retry_iteration)
                                new_key_list[ind] = key_part1
                                new_key = ''.join(new_key_list)
                                for i in range(len(deciphered_sequences[0])):
                                        for seq in deciphered_sequences:
                                                if i < len(seq):
                                                        new_result_text += seq[i]
                                        new_result_text += '|'
                                print(new_result_text)
                                print('\nPossible key:', new_key, '\titeration:', retry_iteration)
                                print('Check if the text is OK and change the letter of the key if needed')
                                print('\nChoose an option:')
                                print('1. Retry')
                                print('2. Finish and go back')
                                des = input()
                                if des == '2':
                                        key = new_key
                                        break
                                if des == '1':
                                        retry_iteration += 1
                if option == '2':
                        print(vigenere_decrypt(text, key, alphabet_dict))
                if option == '3':
                        print(key)
                if option == '4':
                        filename = input('Enter the output file name: ')
                        with open(filename, 'w', encoding='utf-8') as decrypted_file:
                                decrypted_file.write(vigenere_decrypt(text, key, alphabet_dict))
                        print('Decrypted text written to ' + filename)
                        break

def create_IC_csv(IC_dict, filename):
        with open(filename, 'w') as of:
                for key, value in IC_dict.items():
                        of.write('{},{}\n'.format(key, value))
```

```
def main():

	global ALPHABET
	global ALPHABET_DICT
	global KEYS_DICT
	global THEORETICAL_FREQUENCIES

	plaintext = import_data('TEXT_parsed.txt')
	ciphertext_v11 = import_data('ciphertext_var11_parsed.txt')

	vigenere_encrypt_lab('TEXT_parsed.txt', KEYS_DICT, ALPHABET_DICT)

	ciphertext_2 = import_data('TEXT_parsed_encrypted_keylen_2.txt')
	ciphertext_3 = import_data('TEXT_parsed_encrypted_keylen_3.txt')
	ciphertext_4 = import_data('TEXT_parsed_encrypted_keylen_4.txt')
	ciphertext_5 = import_data('TEXT_parsed_encrypted_keylen_5.txt')
	ciphertext_9 = import_data('TEXT_parsed_encrypted_keylen_9.txt')
	ciphertext_12 = import_data('TEXT_parsed_encrypted_keylen_12.txt')
	ciphertext_15 = import_data('TEXT_parsed_encrypted_keylen_15.txt')
	ciphertext_20 = import_data('TEXT_parsed_encrypted_keylen_20.txt')

	texts = {0: plaintext, 2: ciphertext_2, 3: ciphertext_3, 4: ciphertext_4, 5: ciphertext_5,
	9: ciphertext_9, 12: ciphertext_12, 15: ciphertext_15, 20: ciphertext_20}

	indexes_of_coinsidence = {key: calculate_index_of_coinsidence(text, ALPHABET_DICT) for key, text
in texts.items()}

	theoretical_ic = sum([p*p for p in THEORETICAL_FREQUENCIES.values()])

	#for i, j in indexes_of_coinsidence.items():
	#	print('{:>2} {:.6f}'.format(i, j))

	break_Vigenere(ciphertext_v11, ALPHABET_DICT)
	#create_IC_csv(indexes_of_coinsidence, 'myIC.csv')


main()
```