Міністерство освіти і науки України Національний технічний

університет України «Київський політехнічний інститут імені

Ігоря Сікорського» Фізико-технічний інститут

ЛАБОРАТОРНА РОБОТА №3

з дисципліни

«Криптографія»

на тему: «Криптоаналіз афінної біграмної підстановки»

Виконали:

студенти 3 курсу ФТІ

групи ФБ-72

Катрич Дар’я, Марісов Микола

Перевірили:

Чорний О.

Савчук М. М.

Завадська Л. О.

Мета роботи:

Набуття навичок частотного аналізу на прикладі розкриття моноалфавітної підстановки; опанування прийомами роботи в модулярній арифметиці.

Варіант 11:

оквкпкящсройюфчвфбчллфэйлзщоыифуххгьижфбчбжройэжиавкхбоаэлбзьздблфюжвпыхожеуфыхьфисццоисццикшхгтчьюб
рйэунемкщхлфэсццикоэйсыфляьэсблавххуоаебвщвцззабюжэйзэсюфцхцчьдвкьбивцкьхбвхщзфийамсьэхьжофшнйсбгежо
эзхбнннкндхбххьюэкублфйлщзхкгсебэуяффдзэсццоьзкжвхыфиамсьэтцщугтбйрипйпьуфптшьюуйукьлкуеафтбгфмсмешз
чеюцнэпфиздббюакличуьлчяюеьххущьафюешзхксищфнлазьзьфююшлйапзгзхбфйсбцрптозкендзэнлбкнкщуюжбйдрптцьдь
жьяжчэщлакээйфбюыхожобьэгройюфеыхцылеимкжфлйутдяйиакютэффцйабгнйакбхеыкжцлнььфьюдфбгьишллаяфеяфзьзжц
азпзфижжнлмккцниулбргзхбтшяюйцуфьюпзмкабвхщугтбйлзтцсуяазяфыюедяьамсчежоакютэффцшжчядбоапздлчьпдкьуа
вкййлцыхяфнлвцаинарйлзщомксйуккзхьлзтцсуяамсчеылууезкзозуффыоайнтббйцилбвзщьцупзоулцафэйакюсзкшавкаф
эйоендхкыхзннложшушьмйыжбйоеахуиюиазезждрлрйозвфьюяжсблзтцсубйчэбйфувккьбиыжмхбыотмхфыхцылкцбьмуксяф
элойэлтапзсфбззфйузядзозцьндкьдядзюцнэпфизчбющнкщугтбйяфюбфбшуэлтбыхожлфюжйсизсбрйэуэлсымсчеббвкслак
лийияффшлаэлкцзьлсйуяфжнбкдбьэээеипуэлойчьшзкфяфицнкяилклзтцсуожоымехцэжшубкмьцфэйхьазцодбжройэщщрой
вмехщльзйжчячбэлрйычицхьхихкрырйюжкжтуэлтбицззщубгыфхцщбщьзцрипймкжфлйикхкзцьхэлнктцчфлцждефэйоемазя
лвбйццакжтшьдфшулэсбоебббкввещьэкеьюфукззщонихбмсйснгчьфшйсжртджапбляэфббозэиылшжохюжебиисйпчфзикозыв
мсойыцждяуяьрипйшкпзхбвхшлюгеыбйщуэлойактфнльжвхшьткхбшушьщыэфйуяфочшьцьдьсиртмкауэлойуиазезхкрилснг
ьлюджтнйыфьюшкябпфщжэлохщлыфизббвкейукыцйяпфлхмсоймкяждзацаьщьюцсйюжбкьэксюечбшуйудзфисццомкгзбкжинк
чавктккхкуилкххнкуийиаьжнсдмуфпсамсьхоыюзбицьохзнтбшьвфьлшзчебкфичбяфламсьхоыыжбйяфзннкозьхэлойэлвжшь
сьхкчьчьскшьюкщэшьнджодбчясгэфнлсьхбсетцгьпйуклкцзнкшьюкщэшьуяфшебблшжчьтяпфзийияфэсрйсфебгрчлчяфыяф
элтбвжчьтяпфукжфнезфжпфжвхюкщугтбйяфшлкцокшьюкщэшьнджоахакдйукмкхбожойшьазилеьпрпффцехпфцзткмущьщуыж
хеылхккььомажрпзэжюкейэфзкэтеымехцэжнкчуюжьюсйюжххшбжсюжрйгфнлмкактфожщлнйожткхбеххзьзкзткшлдяазмзби
шлицхьсбоемкфждзацшлзилзыхгтуябкоквкпкйцпаюгбкйишьщзпзхббйххптебьктширтйутдньвйгтяавкниеььяозшльщнк
фшвхгтмкбькцрлеьгзттлвххффэйсйюжххцтйуйувздбфыпзвизсрйцзычнйыюсьыэхьжомкчьюдьххсуяфзиоуфукфжсбщулэтб
рйэумсьийиозэррйегдяозфиззикозшьюкюдчбьэыфйсюжрйгфпфщлмкактфпфьжщлнйьиозщжэлнкаыйсюжрйгфарчжжппуехюж
иьщжэлнкаыдяньлзычжспфщлсьхбсдщлеьхиуиакнктдицззкабйыжлзулфжэлчьцньйишьхцпфнлззлбфыйннказезньпьтжкс
ягэуйушжщлнйэлшзвлэуфыхьщупзсцдбсхйуюжэфэйукьфеяебнйгфоафымелзюфлэыфцийидксыоавкчлфихкроюдьхожбйэфчя
зэнисдеьактфххмсфиакюспфзнсыяфщлыдсйпблкцззроблячжяхазслеижщнкикцьаклсюжпфбкикохбыыццзчкшлдящуехнетк
шлыфчэбйвпроцьохзнбйбйэфюенкозьхшвьэеуьзкзчксйюжххдяэфсыюжехгжцллфсепдщйукюдазцоозпащьгикьюгтайндяйл
вхшчицьзуехбхнтбознкооакжфдябкэфйнщпозеьлкцзиоюдпавклижочбфбйудзлхйуюбющнкзффзхфптэсожсюткьхэлюжщлжо
цзеьуинйссыжезеяшьюкзфебэуйуйлыхдзсягхчьюбшуйуезхьчьзщнкцьтцгьсбоееьакюидьхьсбхбщьщзчиыжчидунйдрттэс
ьвозулйафбзяхннкдяьахипазнойоаяфпбэуйулфсыюжикыйдяйлфцждщьэкеьщбюгтаююпаксьгиищзпзхбпфбллкыхсбгфчяыф
пбйуеятыньцьяисцыбюжщзйжьющьжжксягфйэфяьпмгьижмхбыцьшзыйяафбщфэйпфэлзяткфсзвткэрйсяьшлсбчжэфюжебяфхх
кьхцньежвхшцтцокфуфклбшсяаюжбйойодслсьхбизкннлжосыехщлхфюжэлщйцухцньяфьиьтощнкиккннлжовяцлнкэкрлчкхи
слшжщзйжьющььтяарйикхьыжщзйжьющььтэскьшчебфйсбшьзээфсцтцвцикоошжшзхбьвгзфьакюсэсбящьюлкеыллйукыцйяпф
ыццзьлгзхавкхбкйцхьвфбмеэыывкьпущннйюжхунлисьвбказфидялзщоцьжфдящжтттцызеьуишзоочьщбфцщуеясббйньлзпз
кьлкххнкюыткфсгхюювгэфзкфжвхпумаехбюгзоьпмгьшьвкьюдкьхкфбсьщфвзфкзлуиеьылйалцйиыьтжезюсцрщлмкактфыю
шлсыхьшьззфцяфотмхеыыжбйфкэкщьникьылнтнлждзырылзнкисбйхужнююнкиячвьхьлкцгикьылууцьфькьбссбткгдьхьлгь
ыхыжбйоечбюжмтчжфзвцлзьзяцждяукжжнвхчяфкшамспьэикьылуудцуфбюакгапфиишьазщьцсицакикфисццортчкезлзрлкц
зьчуйухбххцуыжпазяэфжпеавкййдзтбрйфцщбьэртфимахичьясзяоттдазазезооскэимйлйоенквффыыфвхбйакнктдсжкстб
чбфбдясдьжщьцзщувзлиждщьэывкейыфдичбдзикхбщььзыискчьцньроьхожойгфбюмкюквкхзюклккякбехцзхбткккьнашупф

ьжоксихкгежошлткиыххлэюснйчэбйеявхфжюзэфхбццшзябдиозмснгьжгзйфдзлзпзткнкюсфзиклирптзнйпчюжбклощхщноз
хкмаобюеропгнджоикпзтоозшсщнфкзльэщлнкззфидящьвггдхикьйиакхкбииннлбияждзьясгэфцлхкьфющнкндйиэигдкевк
хбсдлавкбиьфцуэфйуяфочакябсхбясгэфшфщнбйыжяхьлюжчжебблеиззпзфсщлпбрсозюибкззасвкбьфбйлеьхкхбиипяьэлз
йфьюмккцшзабгдкклырймсебйубюгдчкшгяюмайнвхозэгяюнквфюжозикшсххойчярйфилйукыцйяпфпафбтуюеромклхнйисзя
лзбиюлехойьлоифыкцгиьзообкохзнбйксццрфэсццаьбкьхыжксукичебфбсьмцлкчуйичьпршснкьхжройсхйнсдэннлыжакуф
ебкжбгньпьгьиявхжогьгишькннлжопыфбюжяавкэхманйгзоьпмгььйукыцйяпфмкгцлзюфяапбшьшчебзмлфнкшлчьцьньсыьэ
ыфзкхццзежыфйснйэфвхгтиккьхикьылэлткеякссбляфкфипьяиозяжнеияфбхиьзяфксвхббоесыьэыфебьэщлчяпфщлткчллк

ыхтбукшьмкнкыльзньшьлзоежонцябюжьзщзьоююмкшьюкуеулщфюжюзжйебсехьакозвфзясгэфеыфбюжяьхббйеяцлсьхбйиаз
хункщулэсбгзжрвкроюдгъхбвхтбвщбисьидьфшвозмйптсбфимсшьфцфцсущннйюжхулфларйьэыфэлюжебблшжеьуийснйеыэл
мктцэкэцдбяфожвхбкозмкбхйстбожфбгзоьпмгьхцмсэфбхйсэфцуйубюмкоопдщлйиеьпршссбьхмсрйбьжооефшчвфбгфйснй
мяюзикмкоквкфитжсыэлжрхбазйфбюгьгиозбкдбшвсйюжгзоьпмгьхццзозсйьгбйяфлиуцейыфвзсллкыхнйчьифпфсыюжбйтб
оемкнкхкзжфбеькгвцчьмктцгъхкхццзгзбьшьмккцбквцьзикжрткхкщйебляьияфйлждзьубрйэушуйуюбфбфбхиясиавкохзн
бйчбфбяфнлззюясгэфцлсьхбсешьмхбыдзббвкйскьщьщутдрыньцьяфкжкснйкбпчяфмавкбклийирощжтттцызнквфехгтбклж
сбсдхучяюгуилхбгыфсдьжозмкхбшэеяйньлюдшххнююойсыяфэуттйеифкжоайлчьйирошьцзхбщфвпохьгбыххлэнлжрйуыжез
ткпяьэсджаехмсбиокщоакьхшущьодкьйироткежехмаюжртыикьнтеымеодкьйирыххлэззфжвхзябйннягхуежйсьжйнуиэфщл
акооакацынэфмкшзчйойэфэсехьэнзмкнкуфвхюгтбнйгфоамсбгйимкцйшэзиэлеьбкжийиромкежягчеяигзоьпмгьшахисйнг
бйоемаяфлйчбюжгзззбихжезруозацткежехикхкиясгэфцлсьхбсееьакдяьэбкндхкндйлждзьжогзхфеяжфдяткшйхузвфичь
актфдршсвжнйцупфгидбжркйехгтбкикхкчкчэзэбьщоьжцлшудзэфртаксбехюжбксбоапзтджихбдзчьцьхкроеьтжщпулахдз
лкцхзясдптрйфбьлюдлазззждазйинибкрлсьсуцихбшедьщбщугтйнсдгрфпхьпьгьазйичьлгыцюкмйсбткпзйфзясхждззулгр
ламсбкпащлыффцйидклфбзнквфехттюжюекцшькарйбккцфиьжфбткнкшьозжрпбшьууйугзнкмкхаждазпуежвхвяьлмкактффы
меозфжмстккфыцждфьэшшвоечкфьифцрнйжртдидяябйннягхуьлйсфвхбххцуюжмкхббюьзчьцхзяакщугтбйгзнеазцоезикжф
язтзозфуфпсьжясгэфйлыцылыцнамсчеуырйюжгкябьэщлнклйьфрткжмсткпзпьэйюдбзфуфпниубфэщлчямкиннлпфгтэсазгз
жяозщьякцхсбизльдсюечкшлдяцьндмефшсбрйакуфлавкафэйоефкбкгежофбуихцылщфяамснгфцйишььхэфщлньдяятожыхйу
фкхбпфюжойнлпцшушьвйзсшжфизфеыххлэмксйнгсюгзяфсывкнуэлчьцньяффзсйыщеьгьгишьыжьхожтцеьчьууйуюдикткхи
иолкцзэробькшьуябзыхьэебрссбюючяьэсбяфяапзбхебтббленхичьафюжождфбгцифшдяьэбкщщнкьясгэфщлюжезшзщлсьхб
изсльхпбебблскулшзабьльхпбебблшжфбюдщьэкеьюгблгрщфчэбйбзиоожакычхуяфдкьфэсехазиооемазяпзиочбненкюсцр
шсшснйебвжгзуцщщнкэррйегбютшцуьлеьхцкннлзйыжбювзтзщулэеяозгзйннклфнльзщзьоцьэизфблчязффзиочьфцшьяжйс
сбоеюктжездоббщьйамсфигьсссбююактяфпшвцьрбщьакнумкхкжиозлзйенкиккьафцуфылццзозбискхбжпромсшьчкхбгхак
ккжфнликнкфубюэфэлуинбоаебшбьэыфэскифыехмсмкхкжоыжньфуйлкзаьжоьлвхгрвкмкнихбтбсещьжжчэбйвпропбрйэуяф
яазяэфжпьзкзчьзжйссбоешьесзягзшуйухэыжббйуткнкфжсбляозчьзжездблцшзхбтбоербткмясгэфгкшьйлыхдзшьчкшлдя
эфщнукззщобьоозфщнтбуфыфхуфыхьсцлзгслэлаэллкыхсбэуйуяфыцхбвкыфбйэфяасчнйюжбйфзикушхуцигдййьибузяшьюд
зкьйчэбймахиапвикьылшжсбвхафртшсяфэлойцпхьцьдьяясгэфдибьпзсдрусгеьбкеннкьхмсжпулскезщодфыцйиыьжаххшэ
йуфбехфыцкйфцуфыхьозжилзжячэтццьтквкцзйфбяьлжрйучяиигьоонисучяэфьжксеяакваквкцурйцтхцньаксуьицьацыйту
бйыцежвхююаьщьбььиьтеыххлэнлждзьылыгехьваф чэбккзхкваыфбявхбкйсфвпфизьхдрнешсбйхушвхьгзуиюрвхсыюжцьрб
щьыдьлэдеьдягъгихцтцгъпйукыцйяпфшльжшзхбукшавкафэйоемкфсбкбктоонкнксыехгтсцсихкхьгзакисццикмузяшььз
юбфбоаньчьылыцэщнкюдэзффэйгзоьпмгьижкйптюертохюжебиимкэзчуидхцгьоониюыехэуехжсбйббифкжгирокамснгфцоз

хфсбшлиймачжбйяфшлхуяфззбкакуныфэлыцубгрлахиафойэлнннкфисццохбщьчфдзщьюлчьэиуфщлшфшубкгьоониюбрйэутт
бйыжбйоесйптчяиивцулксжцозикшьохюжебойукыцйяпфпьуфптшьйайндякжоымехцэжшубкежвхтыххлэцркйгхнлдьцфббшф
нльзвояфмкоцшлфжюжбйчьзжрйэлецозикшлеьйиодэкнуфыхьпьэйбиокщоакшлиймачжюешьакозатцркйоекзпашубюсдслбк
лкьзклбцхбойоемкгзхбфисзмкюкэзчуаьщьюцртебьиисфвпфвзэлнквксфюжобщфэйццфцяь

Ключ	Індекс Відповідності	Ключ	Індекс Відповідності	Ключ	Індекс Відповідності
(807, 736)	0.0385289	(438, 357)	0.0380031	(509, 940)	0.0387152
(125, 519)	0.0388723	(836, 799)	0.0383912	(231, 537)	0.037192
(944, 68)	0.0378089	(452, 378)	0.0401079	(427, 752)	0.0380191
(534, 566)	0.039305	(779, 729)	0.0389522	(523, 961)	0.0393731
(154, 582)	0.0395034	(17, 289)	0.0396884	(182, 589)	0.0389828
(514, 378)	0.0390865	(447, 940)	0.0373893	(730, 781)	0.0381708
(351, 357)	0.0377582	(258, 109)	0.0468863	(357, 931)	0.0370528
(318, 83)	0.0381194	(610, 708)	0.0389874	(738, 187)	0.0378237
(412, 628)	0.0376478	(703, 956)	0.0555376		

Розшифрований текст ключем (703, 956):

хорошосэрбиллнехотясунулденьгивкарманвотчтобиллвыпростопосеетеэтуновуютравукогданибудьвдругойразкактолькояп
омрунадругойжеденьможетеперекопатьэтучертовулужайкунукакхватитувастерпенияподождатьещелетпятьшестьчтобыста
рыйболтунуспелотдатьконцыужбудьтеувереныподождусказалбиллсамнезнаюкаквамобяснитьнодляменяжужжаньеэтойко
силкисамаяпрекраснаямелодиянасветевнейвсяпрелестьлетабезнееябыужаснотосковалибеззапахасвежескошеннойтравыт
ожебиллнагнулсяиподнялземликорзинкуяпошелковрагувыславныйюношаивсепонимаетеяуверенизвасполучитсяблестя
щийиумныйрепортерсказалдедушкапомогаяемуподнятькорзинкуявамэтопредсказываюпрошлоутронаступилполденьпосл
еобедадедушкаподнялсяксебенемногопочиталуиттиераикрепкоуснулкогдаонпроснулсябылотричасавокнавливалсяяркийи
веселыйсолнечныйсветдедушкалежалвкроватиивдругвздрогнулслужайкидоносилосьпрежнеезнакомоенезабываемоежуж
жаньечтоэтосказалонктотокоситтравуноведьеетолькосегодняутромскосилионещепослушалдаконечноэтожужжиткосилка
мернонеутомимодедушкавыглянулвокноиахнулдаведьэтобиллэйбиллфорестервамчтосолнцеудариловголовувыкоситеуже
скошеннуютравубиллподнялголовупростодушноулыбнулсяипомахалрукойзнаюнокажетсяутромяработалнеоченьчистодед
ушкаещедобрыхпятьминутнежилсявкроватиислицаегонесходилаулыбкаабиллфорестервсешагалскосилкойнасевернавост
окнаюгинаконецназападиизподкосилкивеселобилдушистыйзеленыйфонтанввоскресеньеутромлеоауфманбродилпосвоем
угаражусловноожидаячтокакоенибудьполеновитокпроволокимолотокилигаечныйключподпрыгнетизакричитначнисменян
оничтонеподпрыгивалоничтонепросилосьвначалокакаяонадолжнабытьэтамашинасчастьядумаллеоможетонадолжнаумещ
атьсявкарманеилионадолжнатебясамогоноситьвкарманеодноязнаютвердосказалонвслухонадолжнабытьяркойлеопостави
лнаверстакбанкуоранжевойкраскивзялсловарьипобрелвдомлинаонзаглянулвтолковыйсловарьтыдовольнаспокойнавесела
ввосторгетебевовсемвезетивсеудаетсяпотвоемувсеидетразумнохорошоиуспешнолинапересталарезатьовощиизакрылаглаз
апрочитаймневсеэтоещеразпожалуйсталеозахлопнулсловарьзакакиеэтогрехиядолженцелыйчасждатьпокатыпридумаешь
мнеответскажитолькодаилинетбольшемненичегоненадотычтоженедовольнанеспокойнаневеселаиневосторгедовольныб
ываюткоровыаввосторгемладенцыданесчастныестарикикоторыеужевпаливдетствосказалалинануанасчеттогочтовеселаса
мвидишькакявеселосмеюськогдаскребуэтураковинулеовнимательнопогляделнаженуилицоегопроясниласьтыправалинаму
жчиныкакойнародникогданичегонесмыслятможетбытьмывырвемсяизэтогозаколдованногокругаужесовсемскорояввосене
жалуюсьзакричалалинаятонеприхожуктебесословареминеговорювысуньязыклеотыведьнеспрашиваешьпочемусердцеуте
бястучитнетолькоднемноиночьюнетаможешьтыспроситьчтотакоебракттоэтозначетнезадавайвопросовестьжетакиелюдивсе
имнадознатькакустроенмиркактокакседакакэтозадумаетсятакойипадаетстрапециивциркелибозадохнетсяпотомучтоемупр
испичилопонятькакунеговгорлемускулыработаютешьпейспидышиперестаньсмотретьнаменятакимиглазамибудтовперв
ыйразвидишьлинаауфманвдругзамерлапотянуланосомвоздухвотбедаавсетывиноватонарвануладверцудуховкиоттудаповал
илдымсчастьесчастьегорестновоскликнулаонаиззаэтогосчастьямыстобойссоримсявпервыйраззаполгодаивпервыйраззадв
адцатьлетнаужинбудутугольявместохлебакогдадымрассеялсялеоауфманаужеисследпростылгрохотлязгсхваткачеловекасвдо
хновениемденьзаднемввоздухетакимелькаюткускиметалладеревамолотокгвоздирейсшинаотверткипоройлеоауфманаохва
тывалоотчаяниеионскиталсяпоулицамвсегдабеспокойныйвсегданачекуонвздрагивалиоборачивалсязаслышавгдетовдалеке
чейтосмехприслушивалсякзабавамдетворыприсматривалсячтовызываетудетейулыбкувечерамионподсаживалсякшумнойк
омпаниинаверандеукогонибудьизсоседейслушалкакстарикивспоминаютпрошлоеитолкуютожизниприкаждомвзрывевесе
льяоживлялсяточногенералкоторыйвидитчтотемныевражескиесилыразгромленыичтоегостратегияоказаласьправильнойп
одорогедомойонторжествовалпоканевходилопятьвсвойгаражгдележалимертвыеинструментыинеодушевленноедеревотогд
аегосияющеелицовновьмрачнелоипытаясьизбытьгоречьнеудачионсожесточениемрасшвыриваликолотилчастисвоеймаши
нысловноэтобылиживыеяростныепротивникинаконецконтурымашиныначаливырисовыватьсяичерездесятьднейиночейдр
ожаотусталостиизможденныйполумертвыйотголодатакойвысохшийипочерневшийточновнегоудариламолниялеоауфманс
потыкаясьпобрелвдомдетиссорилисьиоглушительнокричалидругнадруганопривидеотцатотчасумолкликакбудтопробилур
очныйчасивкомнатувошласамасмертьмашинасчастьяготовапрохрипеллеоауфманлеоауфманпохуделнапятнадцатьфунтовс

казалаегоженаонужедвенеделинеразговаривалсосвоимидетьмионисаминесвоясмотритеонидерутсяегоженатожесаманесво
ясмотритеонапотолстеланадесятьфунтовтеперьейпонадобятсяновыеплатьядаконечномашинаготоваасталимысчастливеект
оскажетлеобросьтымастеритьэтичасывнихневлезетниоднакукушкачеловекунеположеносоватьсявтакиеделагосподубогуэт
онавернонеповредитавотлеоауфмануодинврединикакойпользыеслитакбудетпродолжатьсяещехотьнеделюмыегопохорони
мвегособственноймашиненоэтихсловлеоауфмануженеслышалонсизумлениемсмотрелкакнанеговалитсяпотолоквоттакшт
укаподумалонужележанаполунотутегообволоклатьмаионуслышалтолькокакктототриждыпрокричалчтотонасчетмашиныс
частьянадругоеутроедвараскрывглазаонувиделптицонипроносилисьввоздухеточноразноцветныекамешкиброшенныевнеп
остижимочистыйручейилегонькозвякнувопускалисьнажестянуюкрышугаражасобакивсевозможныхпородтихонькопрокра
дывалисьводвориповизгиваязаглядываливгаражчетверомальчишекдведевочкиинесколькомужчинпомедлилинадорожкепо
томнерешительноподошлипоближеиостановилисьподвишнямилеоауфманприслушалсяипонялчтовлечетихвсехкнемуводв
орголосмашинысчастьятакоеможнобылобыуслышатьлетнимднемвозлекухникакойнибудьвеликаншиэтобылоразноголосо
ежужжаньевысокоеинизкоеторовноетопрерывистоеказалосьтамвьютсяроемогромныезолотистыепчелывеличинойсчашку
истряпаютсказочныеблюдасамавеликаншаудовлетворенномурлычетсебеподноспесенкулицоунееточнорозоваялунавполно
луниевотвотонанеобятнаякаклетоподплыветкдверямиспокойноглянетводворнаулыбающихсясобакнабелобрысыхмальчиш
екиседыхстариковпостойтекагромкосказаллеояведьсегодняещеневключалмашинусаулсаулподнялголовуонтожестоялвниз
уводворесаултыеевключилтыжесамполчасаназадвелелмнеразогретьееахдаясовсемзабыляещетолкомнепроснулсяионопят
ьоткинулсянаподушкулинапринеслаемузавтракиостановиласьуокнаглядявнизнагаражпослушайлеонегромкосказалаонаес
лиэтамашинаивправдутакаякактыговоришьможетбытьонаумеетрожатьдетейаможетонапревратитьстарикасновавюношуи
ещеможновэтоймашинесовсемеесчастьемспрятатьсяотсмертиспрятатьсявоттыработаешьсебянежалеешьавконцеконцовна
дорвешьсяипомрешьчтоятогдабудуделатьвлезувэтотбольшойящикистанусчастливойиещескажимнелеочтоунастеперьзажи
зньсамзнаешькакунасведетсядомвсемьутраяподнимаюдетейкормлюихзавтракомкполовинедевятоговасникогоуженетияос
таюсьоднасостиркойоднасготовкойиноскиштопатьтоженадоиогородполотьивлавкусбегатьисеребропочиститьяразвежалу
юсьятольконапоминаютебекакведетсянашдомлеокакяживутаквотответьмнекаквсеэтоуместитсявтвоюмашинуонаустроена
совсеминачеоченьжальзначитмненекогдабудетдажепосмотретькаконаустроеналинапоцеловалаеговщекуивышлаизкомнат
ыаонлежалипринюхивалсяветернизудоносилсюдазапахмашиныижареныхкаштановчтопродаютсяосеньюнаулицахпариж
акоторогоонникогданевиделмеждузавороженнымисобакамиимальчишкаминевидимкойпроскользнулакошкаизамурлыкала
удверейгаражаиззагаражаслышалсяшорохснежнобелойпенымерноедыханьеприбояудалекихдалекихбереговзавтрамыисп
ытаеммашинудумаллеоауфманвсевместеонпроснулсяпозднооньючтотоегоразбудилодалековдругойкомнатектотоплакалс
аулэтотышепнуллеоауфманвылезаяизкроватиипошелксынумальчикгорькорыдалуткнувшисьвподушкунетнетвсхлипывало
нвсеконченоконченосаултебеприснилосьчтонибудьстрашноерасскажимнесынокномальчиктолькозаливалсяслезамиитутси
дяунегонакроватилеоауфмансамнезнаяпочемувыглянулвокнодверигаражабылираспахнутынастежьонпочувствовалкаквол
осыунеговсталидыбомкогдасаултихоньковсхлипываянаконецзабылсябеспокойнымсномотецспустилсяполестницеподоше
лкгаражуизатаивдыханиеосторожновытянулрукуаа

Код програми:

```
#include <iostream>
#include <string>
#include <memory>
#include <fstream>
#include <vector>
#include <iterator>
#include <bitset>
#include <map>
#include <cmath>



std::vector<uint32_t> get_text(std::string_view path)
{

   std::ifstream Stream(path.data());
   std::vector<uint32_t> text_vec{};
   text_vec.reserve(8000);

   uint32_t temp_let{};
   std::string str{};
   while(getline(Stream, str))
   {
      for(size_t i = 0; i < str.length(); ++i)
      {
         temp_let = str[i];
         temp_let = temp_let & 0b00011111;

         //these 2 ifs corrects missmatch between default alphabet with 'ъ' and alphabet used in this lab
         if(temp_let == 28)
         {
```

```
            temp_let = 26;
         }
         if(temp_let > 27)
         {
            --temp_let;
         }

         text_vec.push_back(temp_let);
      }

   }


   Stream.close();
   text_vec.shrink_to_fit();

   if(text_vec.size() % 2 == 1)
   {
      text_vec.push_back(26);
   }

return text_vec;
}

std::vector< std::pair<uint32_t, uint32_t> > CollectingBigrFreq(std::vector<uint32_t> text_vec)
{
   std::vector<std::pair<uint32_t, uint32_t> > bigr_freq(5, {0,0});// {0,0} - std::initializer_list to correctly invoke
constructor
                                                  // how to use std::make_pair() here


   uint32_t bigr_in_text[31][31]= {};//stats

   for(size_t i = 0; i < text_vec.size() - 1; ++i)
   {
      ++bigr_in_text[ text_vec[i] ][ text_vec[++i] ];//collecting stats
   }



   for(size_t most = 0; most < 5; ++most)
   {
      for(size_t i = 0; i < 31; ++i)
      {
         for(size_t j = 0; j < 31; ++j)
         {
            if( bigr_in_text[i][j] > bigr_in_text[bigr_freq[most].first][bigr_freq[most].second]  )
            {
               bigr_freq[most].first  = i;
               bigr_freq[most].second = j;
            }
         }
      }
      bigr_in_text[ bigr_freq[most].first ][ bigr_freq[most].second ] = 0;//deleting the greatest
   }

return bigr_freq;
}


void ObernMod(int32_t& op1, const uint32_t& mod)
{
   if(op1 < 0)
```

```
    {
        op1 = mod + (op1 );
    }
    else
    {
        op1 = op1 ;
    }
}
int32_t OM(int32_t op1)
{
    if(op1 < 0)
    {
        return op1 = 961 + (op1 % 961);
    }
    else
    {
        return op1 = op1 % 961;
    }
}

int32_t ExtendedEuclid(int32_t a, int32_t b, int32_t& u, int32_t& v) //(c) Handbook of Applied Cryptography
{
    if(b == 0){
    u = 1;
    v = 0;
    return a;
    }
    int32_t v_1 = 1;
    int32_t v_2 = 0;
    int32_t u_1 = 0;
    int32_t u_2 = 1;

    int32_t q = 0;
    int32_t r = 0;

    while( b > 0 )
    {
        q   =   a / b;
        r   =   a - (q *   b);
        u   = u_2 - (q * u_1);
        v   = v_2 - (q * v_1);
        a   =   b;
        b   =   r;
        u_2 = u_1;
        u_1 =   u;
        v_2 = v_1;
        v_1 =   v;
    }
    u = u_2;
    v = v_2;
return a;
}

int32_t LinComp(int32_t a, int32_t b, int32_t m, int32_t& u, int32_t& v, int32_t& add)
{

    int32_t x = 0;
    int32_t greatest_common_divisor = ExtendedEuclid(a, m, u, v);

    ObernMod(u, m);
    ObernMod(v, m);

    if( greatest_common_divisor == 1 ){
```

```
        if( ( (a*u) % m ) == 1 ){
            return x = ( u*b ) % m;
        }
        else{
            return x = ( v*b ) % m;
        }
    }
    else{
        if( (b % greatest_common_divisor) == 0 ){
            add = greatest_common_divisor;
            return x = LinComp( (a/greatest_common_divisor), (b/greatest_common_divisor), (m/
greatest_common_divisor), u, v, add);
        }
        else{
            return x = 1111;
        }
    }

}

int32_t BFromKey(int32_t a, int32_t x, int32_t y, const std::vector<uint32_t>& text) // uint32_t -> int32_t
//debug for text.size() even
{
    if( ( y - (x*a) ) > 0 ){
        return ( (y - (x*a)) % (961) );
    }
    else{
        return ( (961) + ( ( y - (x*a) ) % (961) ) );
    }

}

int32_t AFromKey(int32_t dx, int32_t dy, int32_t& u, int32_t& v, int32_t& add, const std::vector<uint32_t>& text)
{
    return LinComp(dx, dy, 961, u, v, add);
}

void Decrypt(const std::map<std::string, uint32_t>& alph, std::string_view path_pt, const std::vector<uint32_t>&
text_ct,
             int32_t keyA, int32_t keyB, std::vector<uint32_t>& legacy_text)
{
    std::ofstream pt_stream(path_pt.data(), std::ios::app);

    int32_t u = 0;
    int32_t v = 0;
    int32_t car = 0;
    pt_stream << "a:" << keyA << ", b:" << keyB << std::endl;

    car = ExtendedEuclid(keyA, 961, u, v);

    u < 0 ? u = 961 + u : u = u;
    v < 0 ? v = 961 + v : v = v;
    (( (keyA*u) % 961 ) == 1) ? keyA = u : keyA = v;

    int32_t curr_bigr = 0;

size_t ij = 0;
    for(size_t i = 0; i < text_ct.size(); i += 2)
    {
        curr_bigr = text_ct[i] * 31 + text_ct[i + 1];//getting ct bigram
        curr_bigr = (curr_bigr - keyB);
        if(curr_bigr < 0){
            curr_bigr = 961 + (curr_bigr % 961);
```

```
        }
        curr_bigr = ((keyA * (curr_bigr)) % (961));//getting pt bigram


        for(auto im = alph.begin(); im != alph.end(); ++im)
        {
           if(im->second == ((curr_bigr - (curr_bigr % 31) ) / 31))
           {
              legacy_text[ij] = ((curr_bigr - (curr_bigr % 31)) / 31);
              ++ij;
              pt_stream << im->first;
           }
        }

        for(auto im = alph.begin(); im != alph.end(); ++im)
        {
           if(im->second == curr_bigr % 31)
           {
              legacy_text[ij] = (curr_bigr % 31) ;
              ++ij;
              pt_stream << im->first;
           }
        }

     }


pt_stream << std::endl << std::endl;

     pt_stream.close();
}
float Htr(std::vector<uint32_t> text_vec)
{

     std::vector<uint32_t> st (31, 0);
     for(auto it = text_vec.begin(); it != text_vec.end(); ++it)
     {
        ++st[*it];
     }
     float H =0.0;
     for(size_t i = 0; i < 31; ++i)
     {
        if(st[i] !=0)
        {
           H += -((st[i]/((float)text_vec.size()))*(log2(st[i] /((float)text_vec.size()))));
        }
     }


     return H;
}
double CalculateIndex(std::vector<uint32_t> cand_pt,   const std::map<std::string, uint32_t>& alphabet)
{
     std::vector<uint32_t> statistics(31, 0);

     for(auto it = cand_pt.begin(); it != cand_pt.end(); ++it)
     {
        ++statistics[*it];
     }

     double Index = 0.0;
     for(auto it = statistics.begin(); it != statistics.end(); ++it)
     {
```

```
        if(*it != 0)
        {
            Index += ( ( 1 / (double)(cand_pt.size() * (cand_pt.size() - 1))) * ( (*it)*(*it - 1)) );
        }
    }

    return Index;
}



int main()
{
    auto const ascii_location_text = "/home/daria/littleDcr3/crypto_cp_3/variants/11.txt";
    auto const out_plain_text = "/home/daria/littleDcr3/out_PT.txt";
    auto const out_plain_text_every = "/home/daria/littleDcr3/out.txt";

    std::ofstream flusher_stream(out_plain_text);
    flusher_stream << "new day -> new try\n" ; //to erase previous texts
    flusher_stream.close();

    std::vector<uint32_t> text_vec{};
    std::vector< std::pair<uint32_t, uint32_t> > text_bigr_st{};
    std::vector<int32_t> hardcoded_rus_bigr{545, 417, 572, 403, 168};
//    legacy_text.push_back(0);
    //26 is missing
    std::map<std::string, uint32_t> alphabet = { {"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}, {"ь", 26}, {"ы", 27}, {"э", 28}, {"ю", 29}, {"я", 30}  };

    //step 1 : getting CT from ../crypto_cp_3/variants
    text_vec = get_text(ascii_location_text);
    std::cout << text_vec.size() << std::endl;

    std::vector<uint32_t> legacy_text(text_vec.size(), 0);
    //step 2 : collecting the most frequent bigrams in CT
    text_bigr_st = CollectingBigrFreq(text_vec);

    //step 3 :
    std::vector<int32_t> atext_bigr_st;
    for(size_t i = 0; i < 5; ++i)
    {
      atext_bigr_st.push_back(text_bigr_st[i].first * 31 + text_bigr_st[i].second);
    }
    //Decrypt(alphabet, out_plain_text, text_vec, 703, 956, legacy_text);
    //return 0;

    int32_t ah, a, b, addi, u, v, dx, dy;
    u = 0;
    v = 0;
    addi = 0;
    dx = 0;
    dy = 0;
    ah = 0;
    float In{0.0};
    for(auto ity1 = atext_bigr_st.begin(); ity1 != atext_bigr_st.end(); ++ity1)
    {
        for(auto ity2 = atext_bigr_st.begin(); ity2 != atext_bigr_st.end(); ++ity2)
        {
            for(auto itx1 = hardcoded_rus_bigr.begin(); itx1 != hardcoded_rus_bigr.end(); ++itx1)
            {
```

```
for(auto itx2 = hardcoded_rus_bigr.begin(); itx2 != hardcoded_rus_bigr.end(); ++itx2)
{
    addi=0;
  if(*itx1 == *itx2) continue;
  if(*ity1 == *ity2) continue;

  (*itx1 >= *itx2) ? dx = (*itx1 - *itx2) : dx = (961 + (*itx1 - *itx2));
  (*ity1 >= *ity2) ? dy = (*ity1 - *ity2) : dy = (961 + (*ity1 - *ity2));


    a = AFromKey(dx , dy,u, v, addi, text_vec);
    if(a != 1111)
    {
        ah = a;
        if(addi == 0)
        {
           b = BFromKey(a, *itx1, *ity1, text_vec);
          // b = b % 961;
          // a = a % 961;

           Decrypt(alphabet, out_plain_text, text_vec, a, b, legacy_text);
           //autoDetection
           //H = Htr(text_vec);
           In = CalculateIndex(legacy_text, alphabet);
           std::cout << "(" << a << ", " << b <<"),another index : " << In << std::endl;
           for(auto it = legacy_text.begin(); it != legacy_text.end(); ++it)
           {
              *it = 0;
           }

           if(In > 0.054 && In < 0.056)
           {
              return 0;
           }




        }else
        {
           for(int32_t k = 0; k < addi; ++k)
           {
              a = ah + (k*(961/addi));
              b = BFromKey(a, *itx1, *ity1, text_vec);
             // b = b % 961;
             // a = a % 961;
              Decrypt(alphabet, out_plain_text, text_vec, a, b, legacy_text);
           In = CalculateIndex(legacy_text, alphabet);
           std::cout << "(" << a << ", " << b <<"),another index : " << In << std::endl;
           for(auto it = legacy_text.begin(); it != legacy_text.end(); ++it)
           {
              *it = 0;
           }

           if(In > 0.054 && In < 0.056)
           {
              return 0;
           }

           }
        }
    }
    addi=0;
```

```
            }
        }
    }
}

    std::cout << "everything should be done, check out_PT " << std::endl;
//   Decrypt(alphabet, out_plain_text, text_vec, 703, 956, legacy_text);

    std::cout << "CT Index : " << CalculateIndex(text_vec, alphabet) << std::endl;
    std::cout << "PT Index : " << CalculateIndex(legacy_text, alphabet) << std::endl;

return 0;
}
```