

CSCI361 Assignment 1

Asjad Athick
4970512
mama158

Part 1

Ctext-1

Cipher text

fk vdfthd rwt d fkefufeswi j l u r j r k q f p k r s q o w i f z r e a y p q w q f p q f t w i
i w v p
j l i r t s i r p f k w c w p b l o r x w j m i r v f i i m o l e s t r v l o h e o f u r w m f p q l k
a s q v d r o r j l q f l k f p f k r b b r t q f u r l k q d r p m l q w k e d r w e r e f k w i i
e f o r t q f l k p w q q d r p w j r q f j r r k r o c y v f i i a r m o r p r k q a s q f k r b b r t q f u r
l k r j f c d q p w y q d w q q d r p s j l b w i i q d r n s w k q f q f r p l b d r w q i l p q f k
q d r
t l s o p r l b w i i q d r w t q f u f q f r p q d w q d w u r q w h r k m i w t r f k q d r s k f u o p r
f k l s o d l j r p w k e l s q l b q d r j a y e w y w k e a y k f c d q r n s w i i y w q w i i
q f j r p
w k e f k w i i m i w t r p v d r k t r v r h k l v k l q f q v w p f k l i e e w y p v f q d l s o

Index of Coincidence (calculated using Krypto)

IC = 0.065

Average = 0.065

Frequency Distribution Graph

```

-> g
d a file name A1-CSCI361.zip. Download this file and unz
of student number and folder number. Go to the folder th
ll find two ciphertext files (Ctext-1 and Ctext-2).
alphabetic cipher and a Vigenere cipher, respectively.
1 and Ctext-2 using the krypto program provided. Th
is a Windows version KRYPTO.EXE but it's only for 32 b
osbox to run this.
you look to break each cipher and why. Justify choic
he monoalphabetic cipher you decide that C and F are
y.
) as appropriate.
cipher
you must cite them in your report.
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
->

```

Steps to break the cipher:

-> f1	-> f2	-> f3
r 54	qf 16	qdr 8
q 43	dr 13	wii 7
f 40	qd 12	drw 4
w 37	fk 11	fur 4
k 31	rp 10	tqf 4
i 27	ii 9	wke 4
l 25	wi 9	drp 3
p 24	fu 6	fkw 3
d 22	kq 6	iiq 3
e 15	ur 6	kqd 3
s 14	wq 6	lso 3
o 13	jr 5	pfk 3
t 13	ke 5	qdw 3
j 12	kr 5	qfu 3
v 11	lk 5	rpw 3
b 9	pq 5	rtq 3
u 8	pw 5	asq 2
y 8	rk 5	bbr 2
m 7	rt 5	brt 2
a 6	rw 5	bwi 2
c 4	wk 5	cdq 2
h 3	ef 4	dwq 2

1. Apply substitutions for highest frequency words according to statistics of English

(A simple C++ program to substitute characters was written, source code in appendix A. Krypto substitution functionality had a glitch).

I	e
II	t a o i n s h r
III	d l
IV	c u m w f g y p b
V	v k j x q z

Substitution Logs

fk vhfqh rwqh kkefufeswi jlurjkt fp krstowifzre ay ptwtftpfqwi iwvp jlirqsirp fk w cwp blo rxwjmirt
vfii molesqr vlod eofur w mfpstk ast vhror jltflk fp fkrbbrqtfur lk thr pmlt wke hrwere fk wii
efortqflkp wt thr pwjr tfjr rkroc y vfii ar morprkt ast fkrbbrqtfur lkr jfcht pwy thwt thr psj lb wii thr
nswktftfrp lb hrwt ilpt fk thr qlsopr lb wii thr wqtftuftfrp thwt hwur twdrk miwqr fk thr skfuropr fk
lso hljrp wke lst lb thrj ay ewy wke ay kfcht rnswiyy wt wii tfjrp wke fk wii miwqrp vhrkqr vr dklv
klt ft vwp fk lie ewyp vfth lso

Replace: r

With: e

fk vdfth rwt d kkefufeswi jlurjrkq fp krsqowifzre ay pqwqfpqftwi iwvp jlirtsirp fk w cwp blo
rxwjmirt vfii molestr vlo h eofur w mfpqlk asq vdror jlqflk fp fkrbbrqtfur lk qdr pmlq wke drwere
fk wii efortqflkp wq qdr pwjr qfjr rkroc y vfii ar morprkq asq fkrbbrqtfur lkr jfcdq pwy qdwq qdr
psj lb wii qdr nswkqfqrp lb drwq ilpq fk qdr tilsopr lb wii qdr wtqfufqfrp qdwq dwur qwhrk
miwtr fk qdr skfuropr fk lso dljrp wke lsq lb qdrj ay ewy wke ay kfcdq rnswiyy wq wii qfjrp wke
fk wii miwtrp vdrktr vr hklv klq fq vwp fk lie ewyp vfqd lso

Replace: q

With: t

...

ik vzicz ewcz ikriuirswf jluejekt ip kestowfiher ay ptwtipticwf fwvp jlfecsfe p ik w qwp blo
exwjmfe viff molrsce vlod roie w mptlk ast vzeoe jltilk ip ikebbectiue lk tze pmlt wkr zewrer
ik wff rioectilkp wt tze pwje tije ekeoq y viff ae moepekt ast ikebbectiue lke jiqzt pwy tztwt tze
psj lb wff tze nswktitiep lb zewt flpt ik tze clsope lb wff tze wctiuitiep tztwt zwue twdek mfwce
ik tze skiueope ik lso zljep wkr lst lb tzej ay rwy wkr ay kizt enswwfy wt wff tijep wkr ik wff
mfwcep vzekce ve dklv klt it vwp ik lfr rwyp vitz lso

Replace: k

With: n

...

in which each inriuirsal jfuejent ip nestoalizer wy ptatipical lavp jflecslap in a qap bfo exajmle
vill mofrsce vfod roie a mptfn wst vheoe jftfn ip inebbectiue fn the pmft anr hearer in all
rioectifnp at the paje tije eneoq y vill we moepent wst inebbectiue fne jight pay that the psj fb
all the ksantitiep fb heat lfpt in the cfsope fb all the actiuitiep that haue taden mlace in the
snueope in fso hfjep anr fst fb thej wy ray anr wy niht eksally at all tijep anr in all mlacep
vhence ve dnfv nft it vap in flr rayp vith fso

Replace: p

With: s

...

in which each individual movement is neutralized by statistical laws molecules in a gas for example will produce work drive a piston but where motion is ineffective on the spot and headed in all directions at the same time energy will be present but ineffective one might say that the sum of all the quantities of heat lost in the course of all the activities that have taken place in the universe in our homes and out of them by day and by night equally at all times and in all places whence we know not it was in old days with our

Replace: j

With: i

...

in which each individual movement is neutralized by statistical laws molecules in a gas for example will produce work drive a piston but where motion is ineffective on the spot and headed in all directions at the same time energy will be present but ineffective one might say that the sum of all the quantities of heat lost in the course of all the activities that have taken place in the universe in our homes and out of them by day and by night equally at all times and in all places whence we know not it was in old days with our

Replace: j

With: v

...

in which each individual movement is neutralized by statistical laws molecules in a gas for example will produce work drive a piston but where motion is ineffective on the spot and headed in all directions at the same time energy will be present but ineffective one might say that the sum of all the quantities of heat lost in the course of all the activities that have taken place in the universe in our homes and out of them by day and by night equally at all times and in all places whence we know not it was in old days with our

Replace: j

With: k

in which each individual movement is neutralized by statistical laws molecules in a gas for example will produce work drive a piston but where motion is ineffective on the spot and headed in all directions at the same time energy will be present but ineffective one might say that the sum of all the quantities of heat lost in the course of all the activities that have taken place in the universe in our homes and out of them by day and by night equally at all times and in all places whence we know not it was in old days with our

Replace: j

With: g

in which each individual movement is neutralized by statistical laws molecules in a gas for example will produce work drive a piston but where motion is ineffective on the spot and headed in all directions at the same time energy will be present but ineffective one might say

that the sum of all the quantities of heat lost in the course of all the activities that have taken place in the universe in our homes and out of them by day and by night equally at all times and in all places whence we know not it was in old days with our

Key after Cryptanalysis

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

Key: waterbcd fghij klmnopq suvxyz

Ctext-2

Cipher Text

hetvead fie wn brkarxwkenz eez oyx zjvwxwkegtxvz qnijpihr ndam mj
phx

siegbr fb lbkyp agh kde lscqtbse kf bx yws uivj rxtvwtxhcu
amxvipmiu

siml ek omlvn rxwlht mlrj th gikww sln lnrrpiv ejulnqj siml kde
iskanmmrh egiicy mlvj il sicagmqad xrvngr lvwt bw ueshvxwnbdvz
egiicy tru iaqmdqm wmjkrwii es xrknoic kde fejo mhzviegx fb
taip wpiirnew xf ie plzpe plzpek xywn mlv ohxik qphr ndivl z prtgv
phxwv sokhj wnw xyen xzvj th kikttxwhqegijo tame siml kde brkanlmku
oy xyaik iolrxwjeog sw bikqeasl sw emfsmwbei iashplpihr fb smiij

Index of Coincidence

```
[-> i  
IC = 0.045  
Average = 0.045
```

This index of coincidence implies a polyalphabetic cipher

```

[-> i 1
IC = 0.045
Average = 0.045
[-> i2
IC = 0.046
IC = 0.041
Average = 0.043
[-> i3
IC = 0.043
IC = 0.044
IC = 0.047
Average = 0.045
[-> i4
IC = 0.047
IC = 0.041
IC = 0.044
IC = 0.039
Average = 0.043
[-> i5
IC = 0.057
IC = 0.069
IC = 0.070
IC = 0.084
IC = 0.077
Average = 0.071
[-> i6
IC = 0.042
IC = 0.040
IC = 0.051
IC = 0.040
IC = 0.043
IC = 0.044
Average = 0.043

```

This suggests the period is 5

The frequency graphs for the five sub alphabets are as follows

```

-> g 0 5
101         return 0;
102     }
103     //validate input string
104     string plaintext = argv[5];
105     vector<int> numericText;
106     for(int i = 0; i < plaintext.length(); i++)
107         if (kMap.find(plaintext[i]) == kMap.end())
108             cout << "Invalid input string\n";
109             return 0;
110     }
111     //decrypt
112     vector<int> result;
113     //flow control
114     if (argv[4] == "-encrypt") {
115         result = encrypt(numericText, keyA);
116     }
117     else {
118         result = decrypt(numericText, keyA);
119     }
120     //base64 encode
121     string resultText;
122     for(int i = 0; i < result.size(); i++)
123         resultText += reverseMap[result[i]];
124     cout << "Input text: " << plaintext << "\n";
125     cout << "Operation: " << argv[4] << "\n";
126     cout << "Key A: " << argv[2] << "\n";
127     cout << "Key B: " << argv[3] << "\n";
128     cout << "Result: " << resultText << "\n";
129     return 0;
130 }
131
132 int main()
133 {
134     int argc = 6;
135     char* argv[] = {"", "A", "B", "C", "D", "E"};
136     run(argv, argc);
137     return 0;
138 }
139
140 //Test cases
141 int main()
142 {
143     int argc = 6;
144     char* argv[] = {"", "A", "B", "C", "D", "E"};
145     run(argv, argc);
146     return 0;
147 }
148
149 //Test cases
150 int main()
151 {
152     int argc = 6;
153     char* argv[] = {"", "A", "B", "C", "D", "E"};
154     run(argv, argc);
155     return 0;
156 }
157
158 //Test cases
159 int main()
160 {
161     int argc = 6;
162     char* argv[] = {"", "A", "B", "C", "D", "E"};
163     run(argv, argc);
164     return 0;
165 }
166
167 //Test cases
168 int main()
169 {
170     int argc = 6;
171     char* argv[] = {"", "A", "B", "C", "D", "E"};
172     run(argv, argc);
173     return 0;
174 }
175
176 //Test cases
177 int main()
178 {
179     int argc = 6;
180     char* argv[] = {"", "A", "B", "C", "D", "E"};
181     run(argv, argc);
182     return 0;
183 }
184
185 //Test cases
186 int main()
187 {
188     int argc = 6;
189     char* argv[] = {"", "A", "B", "C", "D", "E"};
190     run(argv, argc);
191     return 0;
192 }
193
194 //Test cases
195 int main()
196 {
197     int argc = 6;
198     char* argv[] = {"", "A", "B", "C", "D", "E"};
199     run(argv, argc);
200     return 0;
201 }
202
203 //Test cases
204 int main()
205 {
206     int argc = 6;
207     char* argv[] = {"", "A", "B", "C", "D", "E"};
208     run(argv, argc);
209     return 0;
210 }
211
212 //Test cases
213 int main()
214 {
215     int argc = 6;
216     char* argv[] = {"", "A", "B", "C", "D", "E"};
217     run(argv, argc);
218     return 0;
219 }
220
221 //Test cases
222 int main()
223 {
224     int argc = 6;
225     char* argv[] = {"", "A", "B", "C", "D", "E"};
226     run(argv, argc);
227     return 0;
228 }
229
230 //Test cases
231 int main()
232 {
233     int argc = 6;
234     char* argv[] = {"", "A", "B", "C", "D", "E"};
235     run(argv, argc);
236     return 0;
237 }
238
239 //Test cases
240 int main()
241 {
242     int argc = 6;
243     char* argv[] = {"", "A", "B", "C", "D", "E"};
244     run(argv, argc);
245     return 0;
246 }
247
248 //Test cases
249 int main()
250 {
251     int argc = 6;
252     char* argv[] = {"", "A", "B", "C", "D", "E"};
253     run(argv, argc);
254     return 0;
255 }
256
257 //Test cases
258 int main()
259 {
260     int argc = 6;
261     char* argv[] = {"", "A", "B", "C", "D", "E"};
262     run(argv, argc);
263     return 0;
264 }
265
266 //Test cases
267 int main()
268 {
269     int argc = 6;
270     char* argv[] = {"", "A", "B", "C", "D", "E"};
271     run(argv, argc);
272     return 0;
273 }
274
275 //Test cases
276 int main()
277 {
278     int argc = 6;
279     char* argv[] = {"", "A", "B", "C", "D", "E"};
280     run(argv, argc);
281     return 0;
282 }
283
284 //Test cases
285 int main()
286 {
287     int argc = 6;
288     char* argv[] = {"", "A", "B", "C", "D", "E"};
289     run(argv, argc);
290     return 0;
291 }
292
293 //Test cases
294 int main()
295 {
296     int argc = 6;
297     char* argv[] = {"", "A", "B", "C", "D", "E"};
298     run(argv, argc);
299     return 0;
300 }
301
302 //Test cases
303 int main()
304 {
305     int argc = 6;
306     char* argv[] = {"", "A", "B", "C", "D", "E"};
307     run(argv, argc);
308     return 0;
309 }
310
311 //Test cases
312 int main()
313 {
314     int argc = 6;
315     char* argv[] = {"", "A", "B", "C", "D", "E"};
316     run(argv, argc);
317     return 0;
318 }
319
320 //Test cases
321 int main()
322 {
323     int argc = 6;
324     char* argv[] = {"", "A", "B", "C", "D", "E"};
325     run(argv, argc);
326     return 0;
327 }
328
329 //Test cases
330 int main()
331 {
332     int argc = 6;
333     char* argv[] = {"", "A", "B", "C", "D", "E"};
334     run(argv, argc);
335     return 0;
336 }
337
338 //Test cases
339 int main()
340 {
341     int argc = 6;
342     char* argv[] = {"", "A", "B", "C", "D", "E"};
343     run(argv, argc);
344     return 0;
345 }
346
347 //Test cases
348 int main()
349 {
350     int argc = 6;
351     char* argv[] = {"", "A", "B", "C", "D", "E"};
352     run(argv, argc);
353     return 0;
354 }
355
356 //Test cases
357 int main()
358 {
359     int argc = 6;
360     char* argv[] = {"", "A", "B", "C", "D", "E"};
361     run(argv, argc);
362     return 0;
363 }
364
365 //Test cases
366 int main()
367 {
368     int argc = 6;
369     char* argv[] = {"", "A", "B", "C", "D", "E"};
370     run(argv, argc);
371     return 0;
372 }
373
374 //Test cases
375 int main()
376 {
377     int argc = 6;
378     char* argv[] = {"", "A", "B", "C", "D", "E"};
379     run(argv, argc);
380     return 0;
381 }
382
383 //Test cases
384 int main()
385 {
386     int argc = 6;
387     char* argv[] = {"", "A", "B", "C", "D", "E"};
388     run(argv, argc);
389     return 0;
390 }
391
392 //Test cases
393 int main()
394 {
395     int argc = 6;
396     char* argv[] = {"", "A", "B", "C", "D", "E"};
397     run(argv, argc);
398     return 0;
399 }
400
401 //Test cases
402 int main()
403 {
404     int argc = 6;
405     char* argv[] = {"", "A", "B", "C", "D", "E"};
406     run(argv, argc);
407     return 0;
408 }
409
410 //Test cases
411 int main()
412 {
413     int argc = 6;
414     char* argv[] = {"", "A", "B", "C", "D", "E"};
415     run(argv, argc);
416     return 0;
417 }
418
419 //Test cases
420 int main()
421 {
422     int argc = 6;
423     char* argv[] = {"", "A", "B", "C", "D", "E"};
424     run(argv, argc);
425     return 0;
426 }
427
428 //Test cases
429 int main()
430 {
431     int argc = 6;
432     char* argv[] = {"", "A", "B", "C", "D", "E"};
433     run(argv, argc);
434     return 0;
435 }
436
437 //Test cases
438 int main()
439 {
440     int argc = 6;
441     char* argv[] = {"", "A", "B", "C", "D", "E"};
442     run(argv, argc);
443     return 0;
444 }
445
446 //Test cases
447 int main()
448 {
449     int argc = 6;
450     char* argv[] = {"", "A", "B", "C", "D", "E"};
451     run(argv, argc);
452     return 0;
453 }
454
455 //Test cases
456 int main()
457 {
458     int argc = 6;
459     char* argv[] = {"", "A", "B", "C", "D", "E"};
460     run(argv, argc);
461     return 0;
462 }
463
464 //Test cases
465 int main()
466 {
467     int argc = 6;
468     char* argv[] = {"", "A", "B", "C", "D", "E"};
469     run(argv, argc);
470     return 0;
471 }
472
473 //Test cases
474 int main()
475 {
476     int argc = 6;
477     char* argv[] = {"", "A", "B", "C", "D", "E"};
478     run(argv, argc);
479     return 0;
480 }
481
482 //Test cases
483 int main()
484 {
485     int argc = 6;
486     char* argv[] = {"", "A", "B", "C", "D", "E"};
487     run(argv, argc);
488     return 0;
489 }
490
491 //Test cases
492 int main()
493 {
494     int argc = 6;
495     char* argv[] = {"", "A", "B", "C", "D", "E"};
496     run(argv, argc);
497     return 0;
498 }
499
500 //Test cases
501 int main()
502 {
503     int argc = 6;
504     char* argv[] = {"", "A", "B", "C", "D", "E"};
505     run(argv, argc);
506     return 0;
507 }
508
509 //Test cases
510 int main()
511 {
512     int argc = 6;
513     char* argv[] = {"", "A", "B", "C", "D", "E"};
514     run(argv, argc);
515     return 0;
516 }
517
518 //Test cases
519 int main()
520 {
521     int argc = 6;
522     char* argv[] = {"", "A", "B", "C", "D", "E"};
523     run(argv, argc);
524     return 0;
525 }
526
527 //Test cases
528 int main()
529 {
530     int argc = 6;
531     char* argv[] = {"", "A", "B", "C", "D", "E"};
532     run(argv, argc);
533     return 0;
534 }
535
536 //Test cases
537 int main()
538 {
539     int argc = 6;
540     char* argv[] = {"", "A", "B", "C", "D", "E"};
541     run(argv, argc);
542     return 0;
543 }
544
545 //Test cases
546 int main()
547 {
548     int argc = 6;
549     char* argv[] = {"", "A", "B", "C", "D", "E"};
550     run(argv, argc);
551     return 0;
552 }
553
554 //Test cases
555 int main()
556 {
557     int argc = 6;
558     char* argv[] = {"", "A", "B", "C", "D", "E"};
559     run(argv, argc);
560     return 0;
561 }
562
563 //Test cases
564 int main()
565 {
566     int argc = 6;
567     char* argv[] = {"", "A", "B", "C", "D", "E"};
568     run(argv, argc);
569     return 0;
570 }
571
572 //Test cases
573 int main()
574 {
575     int argc = 6;
576     char* argv[] = {"", "A", "B", "C", "D", "E"};
577     run(argv, argc);
578     return 0;
579 }
580
581 //Test cases
582 int main()
583 {
584     int argc = 6;
585     char* argv[] = {"", "A", "B", "C", "D", "E"};
586     run(argv, argc);
587     return 0;
588 }
589
590 //Test cases
591 int main()
592 {
593     int argc = 6;
594     char* argv[] = {"", "A", "B", "C", "D", "E"};
595     run(argv, argc);
596     return 0;
597 }
598
599 //Test cases
600 int main()
601 {
602     int argc = 6;
603     char* argv[] = {"", "A", "B", "C", "D", "E"};
604     run(argv, argc);
605     return 0;
606 }
607
608 //Test cases
609 int main()
610 {
611     int argc = 6;
612     char* argv[] = {"", "A", "B", "C", "D", "E"};
613     run(argv, argc);
614     return 0;
615 }
616
617 //Test cases
618 int main()
619 {
620     int argc = 6;
621     char* argv[] = {"", "A", "B", "C", "D", "E"};
622     run(argv, argc);
623     return 0;
624 }
625
626 //Test cases
627 int main()
628 {
629     int argc = 6;
630     char* argv[] = {"", "A", "B", "C", "D", "E"};
631     run(argv, argc);
632     return 0;
633 }
634
635 //Test cases
636 int main()
637 {
638     int argc = 6;
639     char* argv[] = {"", "A", "B", "C", "D", "E"};
640     run(argv, argc);
641     return 0;
642 }
643
644 //Test cases
645 int main()
646 {
647     int argc = 6;
648     char* argv[] = {"", "A", "B", "C", "D", "E"};
649     run(argv, argc);
650     return 0;
651 }
652
653 //Test cases
654 int main()
655 {
656     int argc = 6;
657     char* argv[] = {"", "A", "B", "C", "D", "E"};
658     run(argv, argc);
659     return 0;
660 }
661
662 //Test cases
663 int main()
664 {
665     int argc = 6;
666     char* argv[] = {"", "A", "B", "C", "D", "E"};
667     run(argv, argc);
668     return 0;
669 }
670
671 //Test cases
672 int main()
673 {
674     int argc = 6;
675     char* argv[] = {"", "A", "B", "C", "D", "E"};
676     run(argv, argc);
677     return 0;
678 }
679
680 //Test cases
681 int main()
682 {
683     int argc = 6;
684     char* argv[] = {"", "A", "B", "C", "D", "E"};
685     run(argv, argc);
686     return 0;
687 }
688
689 //Test cases
690 int main()
691 {
692     int argc = 6;
693     char* argv[] = {"", "A", "B", "C", "D", "E"};
694     run(argv, argc);
695     return 0;
696 }
697
698 //Test cases
699 int main()
700 {
701     int argc = 6;
702     char* argv[] = {"", "A", "B", "C", "D", "E"};
703     run(argv, argc);
704     return 0;
705 }
706
707 //Test cases
708 int main()
709 {
710     int argc = 6;
711     char* argv[] = {"", "A", "B", "C", "D", "E"};
712     run(argv, argc);
713     return 0;
714 }
715
716 //Test cases
717 int main()
718 {
719     int argc = 6;
720     char* argv[] = {"", "A", "B", "C", "D", "E"};
721     run(argv, argc);
722     return 0;
723 }
724
725 //Test cases
726 int main()
727 {
728     int argc = 6;
729     char* argv[] = {"", "A", "B", "C", "D", "E"};
730     run(argv, argc);
731     return 0;
732 }
733
734 //Test cases
735 int main()
736 {
737     int argc = 6;
738     char* argv[] = {"", "A", "B", "C", "D", "E"};
739     run(argv, argc);
740     return 0;
741 }
742
743 //Test cases
744 int main()
745 {
746     int argc = 6;
747     char* argv[] = {"", "A", "B", "C", "D", "E"};
748     run(argv, argc);
749     return 0;
750 }
751
752 //Test cases
753 int main()
754 {
755     int argc = 6;
756     char* argv[] = {"", "A", "B", "C", "D", "E"};
757     run(argv, argc);
758     return 0;
759 }
760
761 //Test cases
762 int main()
763 {
764     int argc = 6;
765     char* argv[] = {"", "A", "B", "C", "D", "E"};
766     run(argv, argc);
767     return 0;
768 }
769
770 //Test cases
771 int main()
772 {
773     int argc = 6;
774     char* argv[] = {"", "A", "B", "C", "D", "E"};
775     run(argv, argc);
776     return 0;
777 }
778
779 //Test cases
780 int main()
781 {
782     int argc = 6;
783     char* argv[] = {"", "A", "B", "C", "D", "E"};
784     run(argv, argc);
785     return 0;
786 }
787
788 //Test cases
789 int main()
790 {
791     int argc = 6;
792     char* argv[] = {"", "A", "B", "C", "D", "E"};
793     run(argv, argc);
794     return 0;
795 }
796
797 //Test cases
798 int main()
799 {
800     int argc = 6;
801     char* argv[] = {"", "A", "B", "C", "D", "E"};
802     run(argv, argc);
803     return 0;
804 }
805
806 //Test cases
807 int main()
808 {
809     int argc = 6;
810     char* argv[] = {"", "A", "B", "C", "D", "E"};
811     run(argv, argc);
812     return 0;
813 }
814
815 //Test cases
816 int main()
817 {
818     int argc = 6;
819     char* argv[] = {"", "A", "B", "C", "D", "E"};
820     run(argv, argc);
821     return 0;
822 }
823
824 //Test cases
825 int main()
826 {
827     int argc = 6;
828     char* argv[] = {"", "A", "B", "C", "D", "E"};
829     run(argv, argc);
830     return 0;
831 }
832
833 //Test cases
834 int main()
835 {
836     int argc = 6;
837     char* argv[] = {"", "A", "B", "C", "D", "E"};
838     run(argv, argc);
839     return 0;
840 }
841
842 //Test cases
843 int main()
844 {
845     int argc = 6;
846     char* argv[] = {"", "A", "B", "C", "D", "E"};
847     run(argv, argc);
848     return 0;
849 }
850
851 //Test cases
852 int main()
853 {
854     int argc = 6;
855     char* argv[] = {"", "A", "B", "C", "D", "E"};
856     run(argv, argc);
857     return 0;
858 }
859
860 //Test cases
861 int main()
862 {
863     int argc = 6;
864     char* argv[] = {"", "A", "B", "C", "D", "E"};
865     run(argv, argc);
866     return 0;
867 }
868
869 //Test cases
870 int main()
871 {
872     int argc = 6;
873     char* argv[] = {"", "A", "B", "C", "D", "E"};
874     run(argv, argc);
875     return 0;
876 }
877
878 //Test cases
879 int main()
880 {
881     int argc = 6;
882     char* argv[] = {"", "A", "B", "C", "D", "E"};
883     run(argv, argc);
884     return 0;
885 }
886
887 //Test cases
888 int main()
889 {
890     int argc = 6;
891     char* argv[] = {"", "A", "B", "C", "D", "E"};
892     run(argv, argc);
893     return 0;
894 }
895
896 //Test cases
897 int main()
898 {
899     int argc = 6;
900     char* argv[] = {"", "A", "B", "C", "D", "E"};
901     run(argv, argc);
902     return 0;
903 }
904
905 //Test cases
906 int main()
907 {
908     int argc = 6;
909     char* argv[] = {"", "A", "B", "C", "D", "E"};
910     run(argv, argc);
911     return 0;
912 }
913
914 //Test cases
915 int main()
916 {
917     int argc = 6;
918     char* argv[] = {"", "A", "B", "C", "D", "E"};
919     run(argv, argc);
920     return 0;
921 }
922
923 //Test cases
924 int main()
925 {
926     int argc = 6;
927     char* argv[] = {"", "A", "B", "C", "D", "E"};
928     run(argv, argc);
929     return 0;
930 }
931
932 //Test cases
933 int main()
934 {
935     int argc = 6;
936     char* argv[] = {"", "A", "B", "C", "D", "E"};
937     run(argv, argc);
938     return 0;
939 }
940
941 //Test cases
942 int main()
943 {
944     int argc = 6;
945     char* argv[] = {"", "A", "B", "C", "D", "E"};
946     run(argv, argc);
947     return 0;
948 }
949
950 //Test cases
951 int main()
952 {
953     int argc = 6;
954     char* argv[] = {"", "A", "B", "C", "D", "E"};
955     run(argv, argc);
956     return 0;
957 }
958
959 //Test cases
960 int main()
961 {
962     int argc = 6;
963     char* argv[] = {"", "A", "B", "C", "D", "E"};
964     run(argv, argc);
965     return 0;
966 }
967
968 //Test cases
969 int main()
970 {
971     int argc = 6;
972     char* argv[] = {"", "A", "B", "C", "D", "E"};
973     run(argv, argc);
974     return 0;
975 }
976
977 //Test cases
978 int main()
979 {
980     int argc = 6;
981     char* argv[] = {"", "A", "B", "C", "D", "E"};
982     run(argv, argc);
983     return 0;
984 }
985
986 //Test cases
987 int main()
988 {
989     int argc = 6;
990     char* argv[] = {"", "A", "B", "C", "D", "E"};
991     run(argv, argc);
992     return 0;
993 }
994
995 //Test cases
996 int main()
997 {
998     int argc = 6;
999     char* argv[] = {"", "A", "B", "C", "D", "E"};
1000     run(argv, argc);
1001     return 0;
1002 }
1003
1004 //Test cases
1005 int main()
1006 {
1007     int argc = 6;
1008     char* argv[] = {"", "A", "B", "C", "D", "E"};
1009     run(argv, argc);
1010     return 0;
1011 }
1012
1013 //Test cases
1014 int main()
1015 {
1016     int argc = 6;
1017     char* argv[] = {"", "A", "B", "C", "D", "E"};
1018     run(argv, argc);
1019     return 0;
1020 }
1021
1022 //Test cases
1023 int main()
1024 {
1025     int argc = 6;
1026     char* argv[] = {"", "A", "B", "C", "D", "E"};
1027     run(argv, argc);
1028     return 0;
1029 }
1030
1031 //Test cases
1032 int main()
1033 {
1034     int argc = 6;
1035     char* argv[] = {"", "A", "B", "C", "D", "E"};
1036     run(argv, argc);
1037     return 0;
1038 }
1039
1040 //Test cases
1041 int main()
1042 {
1043     int argc = 6;
1044     char* argv[] = {"", "A", "B", "C", "D", "E"};
1045     run(argv, argc);
1046     return 0;
1047 }
1048
1049 //Test cases
1050 int main()
1051 {
1052     int argc = 6;
1053     char* argv[] = {"", "A", "B", "C", "D", "E"};
1054     run(argv, argc);
1055     return 0;
1056 }
1057
1058 //Test cases
1059 int main()
1060 {
1061     int argc = 6;
1062     char* argv[] = {"", "A", "B", "C", "D", "E"};
1063     run(argv, argc);
1064     return 0;
1065 }
1066
1067 //Test cases
1068 int main()
1069 {
1070     int argc = 6;
1071     char* argv[] = {"", "A", "B", "C", "D", "E"};
1072     run(argv, argc);
1073     return 0;
1074 }
1075
1076 //Test cases
1077 int main()
1078 {
1079     int argc = 6;
1080     char* argv[] = {"", "A", "B", "C", "D", "E"};
1081     run(argv, argc);
1082     return 0;
1083 }
1084
1085 //Test cases
1086 int main()
1087 {
1088     int argc = 6;
1089     char* argv[] = {"", "A", "B", "C", "D", "E"};
1090     run(argv, argc);
1091     return 0;
1092 }
1093
1094 //Test cases
1095 int main()
1096 {
1097     int argc = 6;
1098     char* argv[] = {"", "A", "B", "C", "D", "E"};
1099     run(argv, argc);
1100     return 0;
1101 }
1102
1103 //Test cases
1104 int main()
1105 {
1106     int argc = 6;
1107     char* argv[] = {"", "A", "B", "C", "D", "E"};
1108     run(argv, argc);
1109     return 0;
1110 }
1111
1112 //Test cases
1113 int main()
1114 {
1115     int argc = 6;
1116     char* argv[] = {"", "A", "B", "C", "D", "E"};
1117     run(argv, argc);
1118     return 0;
1119 }
1120
1121 //Test cases
1122 int main()
1123 {
1124     int argc = 6;
1125     char* argv[] = {"", "A", "B", "C", "D", "E"};
1126     run(argv, argc);
1127     return 0;
1128 }
1129
1130 //Test cases
1131 int main()
1132 {
1133     int argc = 6;
1134     char* argv[] = {"", "A", "B", "C", "D", "E"};
1135     run(argv, argc);
1136     return 0;
1137 }
1138
1139 //Test cases
1140 int main()
1141 {
1142     int argc = 6;
1143     char* argv[] = {"", "A", "B", "C", "D", "E"};
1144     run(argv, argc);
1145     return 0;
1146 }
1147
1148 //Test cases
1149 int main()
1150 {
1151     int argc = 6;
1152     char* argv[] = {"", "A", "B", "C", "D", "E"};
1153     run(argv, argc);
1154     return 0;
1155 }
1156
1157 //Test cases
1158 int main()
1159 {
1160     int argc = 6;
1161     char* argv[] = {"", "A", "B", "C", "D", "E"};
1162     run(argv, argc);
1163     return 0;
1164 }
1165
1166 //Test cases
1167 int main()
1168 {
1169     int argc = 6;
1170     char* argv[] = {"", "A", "B", "C", "D", "E"};
1171     run(argv, argc);
1172     return 0;
1173 }
1174
1175 //Test cases
1176 int main()
1177 {
1178     int argc = 6;
1179     char* argv[] = {"", "A", "B", "C", "D", "E"};
1180     run(argv, argc);
1181     return 0;
1182 }
1183
1184 //Test cases
1185 int main()
1186 {
1187     int argc = 6;
1188     char* argv[] = {"", "A", "B", "C", "D", "E"};
1189     run(argv, argc);
1190     return 0;
1191 }
1192
1193 //Test cases
1194 int main()
1195 {
1196     int argc = 6;
1197     char* argv[] = {"", "A", "B", "C", "D", "E"};
1198     run(argv, argc);
1199     return 0;
1200 }
1201
1202 //Test cases
1203 int main()
1204 {
1205     int argc = 6;
1206     char* argv[] = {"", "A", "B", "C", "D", "E"};
1207     run(argv, argc);
1208     return 0;
1209 }
1210
1211 //Test cases
1212 int main()
1213 {
1214     int argc = 6;
1215     char* argv[] = {"", "A", "B", "C", "D", "E"};
1216     run(argv, argc);
1217     return 0;
1218 }
1219
1220 //Test cases
1221 int main()
1222 {
1223     int argc = 6;
1224     char* argv[] = {"", "A", "B", "C", "D", "E"};
1225     run(argv, argc);
1226     return 0;
1227 }
1228
1229 //Test cases
1230 int main()
1231 {
1232     int argc = 6;
1233     char* argv[] = {"", "A", "B", "C", "D", "E"};
1234     run(argv, argc);
1235     return 0;
1236 }
1237
1238 //Test cases
1239 int main()
1240 {
1241     int argc = 6;
1242     char* argv[] = {"", "A", "B", "C", "D", "E"};
1243     run(argv, argc);
1244     return 0;
1245 }
1246
1247 //Test cases
1248 int main()
1249 {
1250     int argc = 6;
1251     char* argv[] = {"", "A", "B", "C", "D", "E"};
1252     run(argv, argc);
1253     return 0;
1254 }
1255
1256 //Test cases
1257 int main()
1258 {
1259     int argc = 6;
1260     char* argv[] = {"", "A", "B", "C", "D", "E"};
1261     run(argv, argc);
1262     return 0;
1263 }
1264
1265 //Test cases
1266 int main()
1267 {
1268     int argc = 6;
1269     char* argv[] = {"", "A", "B", "C", "D", "E"};
1270     run(argv, argc);
1271     return 0;
1272 }
1273
1274 //Test cases
1275 int main()
1276 {
1277     int argc = 6;
1278     char* argv[] = {"", "A", "B", "C", "D", "E"};
1279     run(argv, argc);
1280     return 0;
1281 }
1282
1283 //Test cases
1284 int main()
1285 {
1286     int argc = 6;
1287     char* argv[] = {"", "A", "B", "C", "D", "E"};
1288     run(argv, argc);
1289     return 0;
1290 }
1291
1292 //Test cases
1293 int main()
1294 {
1295     int argc = 6;
1296     char* argv[] = {"", "A", "B", "C", "D", "E"};
1297     run(argv, argc);
1298     return 0;
1299 }
1300
1301 //Test cases
1302 int main()
1303 {
1304     int argc = 6;
1305     char* argv[] = {"", "A", "B", "C", "D", "E"};
1306     run(argv, argc);
1307     return 0;
1308 }
1309
1310 //Test cases
1311 int main()
1312 {
1313     int argc = 6;
1314     char* argv[] = {"", "A", "B", "C", "D", "E"};
1315     run(argv, argc);
1316     return 0;
1317 }
1318
1319 //Test cases
1320 int main()
1321 {
1322     int argc = 6;
1323     char* argv[] = {"", "A", "B", "C", "D", "E"};
1324     run(argv, argc);
1325     return 0;
1326 }
1327
1328 //Test cases
1329 int main()
1330 {
1331     int argc = 6;
1332     char* argv[] = {"", "A", "B", "C", "D", "E"};
1333     run(argv, argc);
1334     return 0;
1335 }
1336
1337 //Test cases
1338 int main()
1339 {
1340     int argc = 6;
1341     char* argv[] = {"", "A
```



```

-> f 3
kde 4
egi 3
iml 3
lzp 3
mlv 3
rxw 3
sim 3
brk 2
epl 2
fie 2
gii 2
hrn 2
icy 2
ieg 2
ihr 2
iic 2
ikq 2
jth 2
kan 2
lkd 2
mlk 2
oyx 2
-> f 4
siml 3
brka 2
egii 2
eplz 2
giic 2
hrnd 2
iicy 2
imlk 2
lkde 2
lzpe 2
mlkd 2
pihr 2
plzp 2
xwke 2
adfi 1
adxr 1
aghk 1
agmq 1
aiki 1

```

Popular trigrams are analysed in the text. After a few unsuccessful tries, this character sequence was analysed

```

1  hetve adfie wnbrk arxwk enzee zoyxz jvxwk egtxv zqnij pihrn dammj phxsi egbrf blbky paghk
2  delsc qtbse kfbxy wsuiv jrxtv wtxhc uamxv ipmiu simle komlv nrxwl htmlr jthgi kwsl nlnrr
3  pivej ulnqj simlk deisk anmmr hegii cymlv jilsi cagmq adxrv ngrlv wtbwu eshvx wnbdv zegii
4  cytru iaqmd qmwmj krwii esxrk noick defej omhzv iegxf btaip wpiir newxf ieplz peplz pekxy
5  wnmlv ohxik qphrn divlz prtgv phxwv sokhj wnwxv enxzv jthki ktxwh qegij otame simlk debrk
6  anlmk uoyxy aikio lrxwj eogsw bikqe aslsw emfsm wbeii ashpl pihrf bsmii j

```

```

1  hetvead fie wn brkarxkenz eez oyx zjvxwkegtxvz qnijpihr ndam mj phx
2  siegbr fb lbkyp agh kde lscqtbase kf bx yws uivj rxtvwtxhcu amxvipmiu
3  siml ek omlvn rxwlht mlrj th gikww sln lnrrpiv ejulnqj siml kde
4  iskanmmrh egiicy mlvj il sicagmqad xrvngr lwt bw ueshvxnbdvz
5  egiicy tru iaqmdqm wjkrwii es xrknoic kde fejo mhzviegx fb
6  taip wpiirnew xf ie plzpe plzpek xywn mlv ohxik qphr ndivl z prtgv
7  phxwv sokhj wnw xyen xzvj th kiktwhqegijo tame siml kde brkanlmku
8  oy xyaik iolrxwjeog sw bikqea sl sw emfsmwbei iashplpihr fb smiij

```

The following site was used for statistical cryptograms for the english language:

<http://www.cryptograms.org/letter-frequencies.php>

mlv is likely to be "the"

A partial key is tried on the text

?	?	M (12)	L (11)	V (21)
?	?	T (19)	H (7)	E (4)
		T	E	R

```

hearnad men wn intaresteng anz oft ijvestegatez quespon wdat is phe
oregin ob lighp and tde solqtion kf it hws beej repewtedlu atteipted
sith nk othen resuht thaj to crkwd oun lunapic asulums sith tde
potantiah energy thej is orcanizad enegy hewt is desorgwnizez
energy and iaximqm diskrdet es entnopy tde maso moveient ob
they wpeaned to ie whipe whiper thwn the oheet qpon wdich i prace
phese sords wnd then evej to grktesqgeneso thin sith tde intansitu
of thair exlresseon of birmnass of emmovwble rasolupion ob sterj
-> l 5 15
hearn admen wnint arest engan zofti jvest egate zques pionw datis pheor egino bligh pandt
desol qtion kfith wsbee jrepe wtedl uatte ipted sithn kothe nresu httha jtocr kwdou nluna
picas ulums sitht depot antia hener cythe jisor caniz adene ngyhe wtisd esorg wnize zener
cyand iaxim qmdis krder esent nopyt demas omove iento bthey wpea nedto iewhi pewhi perth
wnthe oheet qponw dichl prace phese sords wndth eneve jtogr ktesq genes othin sitht deint
ansit uofth airex lress eonof birmn assof emmov wbler asolu piono bster j

```

The partial key seems to be a good match. The rest of the key needs to be found.

quespon is likely to be "question"

```

-> l 5 15
hearn admen wnint arest engan zofti jvest egate zques pionw datis pheor egino bligh pandt
desol qtion kfith wsbee jrepe wtedl uatte ipted sithn kothe nresu httha jtocr kwdou nluna
picas ulums sitht depot antia hener cythe jisor caniz adene ngyhe wtisd esorg wnize zener
cyand iaxim qmdis krder esent nopyt demas omove iento bthey wpea nedto iewhi pewhi perth
wnthe oheet qponw dichl prace phese sords wndth eneve jtogr ktesq genes othin sitht deint
ansit uofth airex lress eonof birmn assof emmov wbler asolu piono bster j
->

```

This works well because we still need the first 2 sub-alphabets

P (15)	I (8)	M (12)	L (11)	V (21)
T (19)	I (8)	T (19)	H (7)	E (4)
W	A	T	E	R

The key "water" is tried.

```
[-> S -B water  
[-> l  
learned men an interesting and oft investigated question what is the  
origin of light and the solution of it has been repeatedly attempted  
with no other result than to crowd our lunatic asylums with the  
potential energy then is organized energy heat is disorganized  
energy and maximum disorder is entropy the mass movement of  
they appeared to me white whiter than the sheet upon which i trace  
these words and thin even to grotesqueness thin with the intensity  
of their expression of firmness of immovable resolution of stern
```

The cipher is successfully broken.

The plaintext is:

learned men an interesting and oft investigated question what is
the
origin of light and the solution of it has been repeatedly
attempted
with no other result than to crowd our lunatic asylums with the
potential energy then is organized energy heat is disorganized
energy and maximum disorder is entropy the mass movement of
they appeared to me white whiter than the sheet upon which i trace
these words and thin even to grotesqueness thin with the intensity
of their expression of firmness of immovable resolution of stern

The cipher key is **WATER**