Key Recovery: Monoalphabetic Substitution

Problem Statement: The assignment is about recovering the key that was used to encode a given paragraph of text, given the encoded paragraph and a dictionary of words which the paragraph is a subset of.
The encoding done in this assignment is known as monoalphabetic substitution. This technique involves rearranging the letters of the English alphabet and mapping letters with the new alphabet, which forms the *key* used to encode a paragraph. In this assignment, the key is the rearranged alphabet formed by starting with a 6-letter English word which is followed by the rest of the alphabet picking off from the last letter of this word.
For example:
If the *key* is chosen to be the word **wisdom**, then the new alphabet would look like this:

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

Subsequently, if the text to be encoded (known as the *plaintext*) is “This is my assignment”, then the encoded text (known as the *ciphertext*) would be “Epqc qc vk wccqnxvoxe”.

Task: Given the encoded paragraph and the dictionary `dict.txt`, perform dictionary closure to retrieve the complete key.

Input Format

- The encoded paragraph.

Output Format

- The key used for encoding, as a single string (no spaces between the letters).

Constraints (Assume)

- The English word which begins the key is a meaningful English word, is 6-letters long, contains all unique letters (no repeated letters) and is in the given dictionary.
- Following the last letter of the 6-letter English word, the rest of the key is in alphabetical order looped over from Z-A, excluding any letters present in the 6-letter English word.
- All the words in the decoded paragraphs exist in the given dictionary `dict.txt`.

Example Input 1:
nce yhg sx gqo foc gqo jcgoe sf cf avho cf gqo zogcvf yn gqo vyiovsofg yn ryexnvyjoef, cxb cf rvoce cf gqo rvoceofg pvcff; ahg sg sf ioel booz, boozoe gqcx cxl cxrqye-rcavo rcx eocrq, cxb wcxl rqherq gyjoef jyhvb qcio gy ao zhg yxo yx gqo gyz yn cxygqoe gy eocrq neyw gqo ayggyw yhg yn gqo jcgoe. byjx gqoeo vsio gqo foc zoyzvo.

Example Output 1:
carbonpqstuvwxyzdefghijklm

Example Input 2:
pog gxgojof pe mbvg uwnngt, rgeptg vhg ojogvggovh igovwta hbd tgbihgd pog-vhjtd pe jvu uqbo, b apwof nbo bod ypnbo, vhg mbvvgt ibttajof b ihjmd, ygtg bqqtpbihjof vhg mbtfg xjmmbfg pe ygadpo-qtjptu, jo wqqgt yguugz, po eppv. vhga ygtg qmbjoma rwv opv jmm imbd, vhpwfh vhg vhjil hpbt pe dwuv yhjih hbd biiwnwmbvgd po vhgjt uhpgu bod fbtngovu etpn bo prxjpwuma mpof kpwtoga mgov b djubdxbovbfgpwu uhbrrjoguu vp vhgjt bqqgbtboig kwuv opy.

Example Output 2:
bridgefhjklmnopqstuvwxyzac

Example Input 3:
epox q pwd wxyepob epyfnpe: zpkcqsc dqcnfcec vo w uqeeuo iqe xyh, ife q fcod ey oxryk dyqxn zpkcqsc. hpk dqd q oxryk qe? q fcod ey zuwk hqep qe. q fcod ey dy hpweogob q moue uqto dyqxn-qe dqd xye pwgo ey dy hqep hpoepob qe hwc qvzybewxe myb epo dogouyzvoxe ym xfsuowb zpkcqsc, ife hpoepob qe hwc qxeoboceqxn wxd wvfcqxn myb vo ey zuwk hqep. hpox q hwc qx pqnp cspyyu, q hyfud coo hweob bfxxqxn yfe ym w mwfsoe nbyhqxn xwbbyhob, wxd hyxdob qm q syfud mqnfbo yfe hpwe doeobvqxoc epwe sfbgo. q myfxd qe hwc bwepob owck ey dy. q dqd xye pwgo ey dy qe; qe hwc xye qvzybewxe myb epo mfefbo ym csqoxso; cyvoiydk ouco pwd wubowdk dyxo qe. epwe dqd xye vwto wxk dqmmoboxso. q hyfud qxgoxe epqxnc wxd zuwk hqep epqxnc myb vk yhx oxeobewqxvoxe. cy q nye epqc xoh weeqefdo. xyh epwe q wv ifbxod yfe wxd q hquu xogob wssyvzuqcp wxkepqxn, q pwgo nye epqc xqso zycqeqyx we epo fxqgobcqek eowspqxn suwccoc hpqsp q bwepob oxryk, wxd rfce uqto q bowd epo wbwiqwx xqnpec myb zuowcfbo, q wv nyqxn ey zuwk hqep zpkcqsc, hpoxogob q hwxe ey, hqepyfe hybbkqxn wiyfe wxk qvzybewxso hpwecyogob. hqepqx w hoot q hwc qx epo swmoeobqw wxd cyvo nfk, myyuqxn wbyfxd, epbyhc w zuweo qx epo wqb. wc epo zuweo hoxe fz qx epo wqb q cwh qe hyiiuo, wxd q xyeqsod epo bod vodwuuqyx yx epo zuweo nyqxn wbyfxd. qe hwc zboeek yigqyfc ey vo epwe qe hoxe wbyfxd mwceob epwx epo hyiiuqxn. q pwd xyepqxn ey dy, cy q cewbe ey mqnfbo yfe epo vyeqyx ym epo byeweqxn zuweo. q dqcsygob epwe hpox epo wxnuo qc gobk cuqnpe, epo vodwuuqyx byeweoc ehqso wc mwce wc epo hyiiuo bweo – ehy ey yxo. qe swvo yfe ym w syvzuqsweod oafweqyx! epox q epyfnpe, "qc epobo cyvo hwk q swx coo qx w vybo mfxdwvoxewu hwk, ik uyytqxn we epo mybsoc yb epo dkxwvqsc, hpk qc qe ehy ey yxo?" q dy xye bovoviob pyh q dqd qe, ife q fueqvweouk hybtod yfe hpwe epo vyeqyx ym epo vwcc zwbeqsuoc qc, wxd pyh wuu epo wssouobweqyxc iwuwxso ey vwto qe syvo yfe ehy ey yxo. q cequu bovoviob nyqxn ey w csqoxeqce wxd cwkqxn, "pok, q xyeqsod cyvoepqxn qxeoboceqxn. pobo epo zuweo nyoc wbyfxd cy, wxd epo bowcyx qe qc ehy ey yxo qc..." wxd q cpyhod pqv epo wssouobweqyxc. po cwkc, "epwe qc zboeek qxeoboceqxn, ife hpwe qc epo qvzybewxso ym qe? hpk wbo kyf dyqxn qe?"

"pwp!" q cwk. "epobo qc xy qvzybewxso hpwecyogob. q wv rfce dyqxn qe myb epo mfx ym qe." pqc bowseqyx dqd xye dqcsyfbwno vo; q pwd vwdo fz vk vqxd q hwc nyqxn ey oxryk zpkcqsc wxd dy hpweogob q uqtod. q hoxe yx ey hybt yfe oafweqyxc ym hyiiuoc. epox q epyfnpe wiyfe pyh ouosebyx ybiqec cewbe ey vygo qx bouweqgqek. epox epobo qc wx afweqyx qx ouosebydkxwvqsc. wxd epox afwxefv ouosebydkxwvqsc. wxd iomybo q txoh qe (qe hwc w gobk cpybe eqvo) q hwc zuwkqxn, hybtqxn, bowuuk hqep epo cwvo yud zbyiuov epwe q uygod cy vfsp, epwe q pwd ceyzzod hybtqxn yx hpox q hoxe ey uyc wuwvyc: vk epocqc-ekzo zbyiuovc; wuu epyco yud-mwcpqyxod, hyxdobmfu epqxnc. qe hwc ommybeuocc. qe hwc owck ey zuwk hqep epoco epqxnc. qe hwc uqto fxsybtqxn w iyeeuo: ogobkepqxn muyhod yfe ommybeuoccuk. q wuvyce ebqod ey bocqce qe! epobo hwc xy qvzybewxso ey hpwe q hwc dyqxn, ife fueqvweouk epobo hwc. epo dqwnbwvc wxd epo hpyuo ifcqxocc epwe q nye epo xyiou zbqlo myb swvo mbyv epwe zqdduqxn wbyfxd hqep epo hyiiuqxn zuweo.

Example Output 3:
wisdomnpqrtuvxyzabcefghjkl

File Handling to read the dictionary: As `dict.txt` is a very large file, hard-coding the words in an array is impractical. Consider other approaches to access the words in the dictionary. Feel free to look up various ways. One such method is file handling, which you can use to read in the contents of the file word by word and, if required, store these in an array.
Here is an example of how you can open the dictionary file and store all the words in an array (you can do this in various other ways too :D):

```
string dictionary[97570];
int dictSize = 0;
ifstream file(“dict.txt”);
if (file.is_open()) {
      string line;
      while (getline(file, line) && dictSize < arrayLength) {
            dictionary[dictSize++] = line;
      }
      file.close();
} else {
      cerr << "Unable to open file: " << dictionaryFile << endl;
      return 1;
}
```

Example Solution Strategy:

The solution strategy I can think of in this case would be as follows:

1) compile in an array all the 6 letters words in the dictionary

2) since we've given the key structure, pull out the 6-letter words one by one and construct the key by filling in the remaining alphabet in order excluding letters present in the 6-letter word

3) perform monoalphabetic substitution on the encoded paragraph with the current key, store the decoded paragraph in a new array (basically retain the original paragraph)

4) check if all the words in the decoded paragraph exist in the given dictionary - if yes, terminate and print the key, if no, restart the process with the next 6-letter word.

Submission Instructions:

- Name your code file `solution.cpp` and upload it on Moodle under the relevant assignment.