// First, parse the letter values from the given plate string.The parsed plate string will further be
// passed to findminWord function to find the shortest possible word from the plate string

```
string plateParser(string inputPlate)
{
    string plate;
    for(int i=0; i<inputPlate.length(); i++)
    {
        if(isalpha(inputPlate[i]))
            plate.push_back(inputPlate[i]);
    }
    return plate;
}
```

// the resultant string plate will be passed to findMinWord function here. Dictionary is passed too.

```
string findMinWord(string plate, vector<string> dictionary)
{
     int plateFreq[26] = countFreq(plate); // count the frequencies of the letters in plate string
     string minWord;

     for (string word : dictionary)
     {
        int wordFreq[26] = countFreq( word ); // count the frequencies of the letters in dict. word
        if (isMatch( plateFreq, wordFreq))
            return word; // if fully matching word is found, return immediately
     }
     // if no match for the same size of plate letters, then start adding extra letters
    // arraySum is a helper function returning sum of array elements
        Int k=1;
     while(arraySum(wordFreq)<45) // the longest word's number of letters is 45 in English.
     {
          for (int i=0; i<26; i++)
          {
               plateFreq[i]+k; // add one more letter to the set of plate letters. Start from a to z
               for (string w : dictionary)
               {
                    wordFreq[26] = countFreq( w );
                    if(isMatch( plateFreq, wordFreq[i]))
```

```
                                                return w;
                              }
                              plateFreq[i]-k; // if no match for earlier added letter, remove addition, go to the
// next letter
                              K++; //increase number of added letter to two, three etc until match is found
                    }
          }
}


bool isMatch(int plateFreq[], int wordFreq[])
{
          for (int i=0; i<26; i++)
              if (plateFreq[i] != wordFreq[i])
                  return false;
          return true;
 }


int[] countFreq( string s)
{
          int freq[26] = {0};
          for ( int i=0; i< s.length(); i++ )
          {
              char c = s.at(i);
              if ( ( c - 'a' ) >= 0 && ( c - 'a' ) < 26 ) // assume dictionary is lowercase
                  freq[c - 'a']++;
           }
          return freq;
}
```