Info:

I have attached a dictionary file containing words of the given inputs for testing.

I have also included a “commented” code in program which takes dictionary input from user.

We can use any of the above thing for reference to dictionary.

Program:

```java
import java.io.*;
import java.util.*;

public class LevenshteinPuzzle {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Set<String> dict = new HashSet<String>();

        //dictionary reference for given tests inputs in puzzle.
        try {
            File file = new
File("C:\\Users\\shubham\\Desktop\\Rally_Health_Puzzle\\index.txt");
            BufferedReader br = new BufferedReader(new FileReader(file));
            String st;
            String[] array = null;
            while ((st = br.readLine()) != null) {
                array = st.split(",");
                for (int i = 0; i < array.length; i++) {
                    dict.add(array[i]);
                }
            }
        } catch (Exception e) {
            System.out.println(e.getStackTrace());
        }

        //System.out.println("Please fill the Dictionary First and when done press
enter twice");   // Taking dictionary input.
//        while (true) {
//            String line = sc.nextLine();
//            if (line.equals("")) {
//                break;
//            } else {
//                dict.add(line);
//            }
//        }

        String str = sc.nextLine();
        String[] strArray = str.split(" ");
        int[] array = new int[4];
        //System.out.println("Please enter the Cost of addition, deletion, change and
Anagram");
        for (int i = 0; i < array.length; i++) {
            array[i] = Integer.parseInt(strArray[i]);
        }
        String startWord = sc.nextLine();
        String endWord = sc.nextLine();

        LevenshteinPuzzle levenshteinPuzzle = new LevenshteinPuzzle();
        int result = levenshteinPuzzle.leveDisPuzzle(array, startWord.toUpperCase(),
endWord.toUpperCase(), dict);
        System.out.println("The minimum cost to get from start to end word is : " +
result);
```

```
    }

    public int leveDisPuzzle(int[] array, String startWord, String endWord,
Set<String> dict) {

        int totalCost = 0;

        while (!startWord.equals(endWord)) {

            String returnDelete = delete(startWord, dict);
            String returnAdd = add(startWord, dict);
            String returnChange = change(startWord, dict);
            String returnAnagram = anagram(startWord, dict, "");

            if (returnDelete.length() == 0 && returnAdd.length() == 0 &&
returnChange.length() == 0 && returnAnagram.length() == 0) {
                return -1;
            }

            if (returnDelete.length() >= 3) {
                totalCost += array[1];
                startWord = returnDelete;
            } else if (returnAdd.length() >= 3) {
                totalCost += array[0];
                startWord = returnAdd;
            } else if (returnChange.length() >= 3) {
                totalCost += array[2];
                startWord = returnChange;
            } else if (returnAnagram.length() >= 3) {
                totalCost += array[3];
                startWord = returnAnagram;
            } else {
                return -1;
            }

        }

        return totalCost;


    }

    // Change Logic (Change a Letter)
    List<String> list = new ArrayList<>();
    HashMap<String, Integer> map = new HashMap<>();

    public String change(String word, Set<String> dict) {
        //System.out.println("Starting change function....");
        String nWord = null;
        //HashMap<String, Integer> map = new HashMap<>();
        char[] letters = word.toCharArray();
        loop:
        for (int i = 0; i < letters.length; i++) {
            char hold = letters[i];
            for (char ch = 'A'; ch <= 'Z'; ch++) {
                if (ch != hold) {
                    letters[i] = ch;
                    nWord = new String(letters);
                    if (dict.contains(nWord)) {
                        if (map.containsKey(nWord)) {
                            continue;
                        } else {
```

```java
                            //System.out.println(" wORD IS :" + nWord);
                            map.put(nWord, 1);
                            break loop;
                        }
                    }

                }
                letters[i] = hold;
            }
        }
        if (dict.contains(nWord) && nWord.length() >= 3) {
            return nWord;
        }

        return "";
    }


    // Anagram Logic (Anagram of word)
    public String anagram(String word, Set<String> dict, String prefix) {
        //System.out.println("Starting anagram function....");
        List<String> list = new ArrayList<>();
        List<String> result = getList(word, dict, list, prefix);
        //System.out.println(result);
        for (int i = 0; i < result.size(); i++) {
            if (dict.contains(result.get(i)) && !result.get(i).equals(word)) {
                return result.get(i);
            }
        }
        return "";
    }

    public List<String> getList(String word, Set<String> dict, List<String> list,
String prefix) {
        int n = word.length();
        if (n == 0) {
            // System.out.println(prefix);
            list.add(prefix);
        } else {
            for (int i = 0; i < n; i++) {
                getList(word.substring(0, i) + word.substring(i + 1), dict, list,
prefix + word.charAt(i));
            }
        }
        return list;
    }


    // Delete Logic (Delete a Letter)
    HashSet<String> set2 = new HashSet<>();

    public String delete(String word, Set<String> dict) {
        //System.out.println("Starting delete function....");
        String helper = null;
        for (int i = -1; i < word.length() - 1; i++) {
            //System.out.println(word.substring(0, i + 1) + word.substring(i + 2,
word.length()));
            if (dict.contains(word.substring(0, i + 1) + word.substring(i + 2,
word.length()))) {
                helper = word.substring(0, i + 1) + word.substring(i + 2,
word.length());
                set2.add(word);
                break;
            }
        }
        if (dict.contains(helper) && helper != null) {
            return helper;
        }
        return "";
    }


    // Addition Logic
    public String add(String word, Set<String> dict) {
        //System.out.println("Starting add function....");
        String helper = null;

        //adding letter at first and in between string
        loop:
        for (int i = -1; i < word.length() - 1; i++) {
            for (char ch = 'A'; ch <= 'Z'; ch++) {
                //System.out.println(word.substring(0, i + 1) + ch + word.substring(i
+ 1, word.length()));
                if ((dict.contains(word.substring(0, i + 1) + ch + word.substring(i +
1, word.length()))) && (!set2.contains((word.substring(0, i + 1) + ch +
word.substring(i + 1, word.length()))))) {
                    helper = word.substring(0, i + 1) + ch + word.substring(i + 1,
word.length());
                    break loop;
                }
            }
        }
        //adding letter at last (corner case)
        for (char ch = 'A'; ch <= 'Z'; ch++) {
            //System.out.println(word + ch);
            if ((dict.contains(word + ch)) && (!set2.contains(word + ch))){
                helper = word + ch;
                break;
            }
        }
        if (dict.contains(helper) && helper != null) {
            return helper;
        }
        return "";
    }
}
```