```cpp
#include <iostream>
#include <cstring>
#include <fstream>

using namespace std;

struct elem{
    char parola[31];
    elem *pun;
};

struct Dizionario{
    elem *testa;
};

void inizializzaDizionario(Dizionario &D){
    D.testa = NULL;
}

bool inserisciParola(Dizionario &D, const char* str){
    //controllo stringa NULL
    if(str == NULL)
        return false;

    //controllo lunghezza stringa
    if(strlen(str)>30)
        return false;

    //controllo caratteri stringa
    for(int i=0;i<strlen(str);i++)
        if(str[i] < 'a' || str[i] > 'z')
            return false;


    //controllo duplicati
    for(elem *p = D.testa; p!=NULL; p=p->pun){
        if(strcmp(p->parola,str) == 0)
```

```
            return false;
    }

    //creo elemento e inserisco in testa
    elem *r = new elem;
    strcpy(r->parola, str);
    r->pun = D.testa;
    D.testa = r;

    return true;
}

bool rimuoviParola(Dizionario &D, const char* str){

    //controllo lunghezza stringa
    if(strlen(str)>30)
        return false;

    //si può non controllare se str
    //è fatto di minuscole e basta, ci pensa già la strcmp

    //cerco l'elemento, se lo trovo q punta all'elemento trovato
    //e q punta al precedente
    elem *p, *q;
    for(q=D.testa; q!=NULL && strcmp(q->parola,str)!=0; q=q->pun)
        p=q;

    //prima controllo se non l'ho trovato
    if(q==NULL)
        return false;

    //l'ho trovato, controllo se è in testa o se nel mezzo
    if(q==D.testa)
        D.testa = D.testa->pun;
    else
        p->pun = q->pun;
```

```
    //elimino l'elemento e ritorno true
    delete q;
    return true;
}

int contaIniziali(Dizionario &D, char c){

    int conta = 0;

    //scorro la lista e controllo parola[0] in ogni elem
    for(elem *q = D.testa; q!=NULL; q=q->pun){
        if(q->parola[0] == c)
            conta++;
    }

    return conta;
}

void stampaDizionario(Dizionario &D){

    cout << '<';

    for(elem *q = D.testa; q!=NULL; q=q->pun){
        cout << q->parola;
        if(q->pun != NULL)
            cout << ", ";
    }

    cout << '>' << endl;
}

int* quanteOccorrenze(Dizionario &D){

    //creo array di 26 elementi in memoria dinamica
    //se non lo creo in memoria dinamica il vettore alla fine
    //della funzione viene distrutto in memoria!
    int *arr = new int[26];
```

```
    char c = 'a';

    //richiamo 26 volte contaIniziali, e poi ogni volta incremento c
    for(int i=0;i<26;i++){
        arr[i] = contaIniziali(D, c);
        c++;
    }

    //ritorno il puntatore
    return arr;
}

void esercizio2(const char *str){
    ifstream in;

    in.open(str);

    if(!in) {
        cout << "Errore apertura" << endl;
        return;
    }

    int somma = 0;
    int n;

    //leggiamo finché non arriviamo alla fine del file
    // le condizione !in oppure !in.eof() sono equivalenti
    while(!in){
        in >> n;
        somma += n;
    }

    in.close();

    cout << "la somma dei numeri nel file e' " << somma << endl;
}
```

```
int esercizio3(int* arr, int N){
    if(N == 0)
        return 0;

    if(arr[N-1]%2 == 1)
        return 1 + esercizio3(arr, N-1);
    else
        return 0 + esercizio3(arr, N-1);
}

int main() {

    Dizionario D;
    inizializzaDizionario(D);

    inserisciParola(D, "amo");
    inserisciParola(D, "botte");
    inserisciParola(D, "zattera");
    inserisciParola(D, "botte");
    inserisciParola(D, "Cane");

    stampaDizionario(D);

    inserisciParola(D, "ananas");
    rimuoviParola(D, "botte");

    stampaDizionario(D);

    int n = contaIniziali(D, 'a');
    cout << "numero di a: " << n << endl;
    n = contaIniziali(D, 'b');
    cout << "numero di b: " << n << endl;

    inserisciParola(D, "botte");
    inserisciParola(D, "banana");
    inserisciParola(D, "banca");
```

```
    stampaDizionario(D);
    int *v = quanteOccorrenze(D);

    cout << '[';
    for(int i=0;i<25;i++)
        cout << v[i] << ", ";
    cout << v[25] << ']' << endl;

    esercizio2("input.txt");

    n = esercizio3(v, 26);
    cout << "numero dispari nel vettore: " << n << endl;

    return 0;
}

/* Soluzione esercizio 1
 *
 * Il numero ABCDEF è facilmente scomponibile in binario espandendo ogni cifra in 4 bit
 * A = 1010
 * B = 1011
 * C = 1100
 * D = 1101
 * E = 1110
 * F = 1111
 * Quindi ABCDEF = 1010 1011 1100 1101 1110 1111
 * a questo punto i bit si raggruppano in gruppi di 3 -> 101 010 111 100 110 111 101 111
 * ogni gruppetto di 3 si trasforma in base 8         ->  5   2   7   4   6   7   5   7
 * la conversione in base 8 di ABCDEF è 52756757
 *
 */
```