NOME COGNOME_____MATRICOLA_____

PRE-APPELLO - Leggere con molta attenzione prima di rispondere
TOTALE 16 punti

ESERCIZIO 1 - Vero o falso? [2 punti]

[F_] All'interno della memoria primaria troviamo una porzione chiamata *heap* che memorizza le variabili locali dichiarate nel programma in esecuzione.

[V_] Nei linguaggi di alto livello associamo "nomi" ai dati, in modo da permettere una più facile scrittura e comprensione del codice. Una dichiarazione di variabile è un'istruzione che realizza un'associazione logica tra un identificatore (o nome) e un'area di memoria in grado di contenere un dato di un certo tipo

[F_] All'interno della memoria primaria troviamo solo dati

[V_] La dichiarazione di una costante realizza un'associazione permanente tra un nome e un valore

[F_] Un cast implicito da float a int è safe

[V_] I tipi interi offerti dal C++ descrivono lo stesso tipo di dato, ma considerano diversi domini di valori.

ESERCIZIO 2 [2 punti]

Fornire un'implementatione dell'algoritmo selection sort per l'ordinamento di un array di interi

```
    void selection_sort(int A[], int size) {


    la soluzione la trovate nelle slide IP8_operazioni_su_sequenze.pdf
    nota: riportare un algoritmo di ordinamento diverso da selection sort rappresenta
    una risposta errata (valutata con uno 0)

    }
```

ESERCIZIO 3 [8 punti]

Consideriamo la seguente implementazione di sequenza ordinata di string senza ripetizioni

```
struct codeword {
    string w;
    codeword *succ;
}
typedef codeword* bag_of_words;
```

Assumiamo di avere una libreria di funzioni che operino sulla struttura dati di cui sopra e che comprenda (funzioni primitive che diamo per assunte e che utilizzeremo ogni qual volta ci siano necessarie nel seguito)

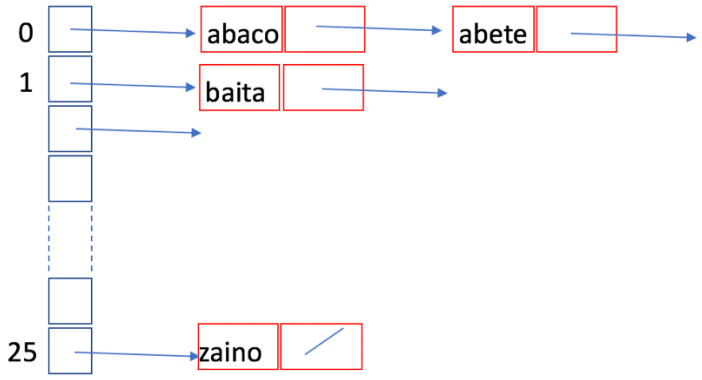
```
bool is_in(string, const bag_of_words &);
void insert(string, bag_of_words &); // insert in order without repetitions
void empty(bag_of_words &);
```

a. Realizzare una funzione che ritorni true se tutti gli elementi di una bag_of_words iniziano per un dato char

```
   bool check(char c, const bag_of_words &b ){ //versione ricorsiva
            if empty(b) return true;
            if (b->w[0]!=c) return false;
            return check(c,b->succ);
      };

   bool check(char c, const bag_of_words &b ){{ //versione iterativa
        if empty(b) return true;
        bag_of_words temp = b;
        while (temp!=nullptr) {
           if (temp->w[0]!=c) return false;
           temp=temp->succ;
        }
        return true; // arrivo qui solo se tutti gli elementi iniziano per c
        }
```

b. Realizzare una struttura dati dictionary, che memorizzi un insieme di bag_of_words secondo lo schema qui di seguito descritto



RISPOSTA:

```
const int SIZE = 26;

struct Dictionary {
        bag_of_words d[SIZE];
};
```

c. Realizzare una funzione booleana che cerchi una parola in un dictionary,
[HINT: Assumiamo di avere a disposizione una funzione ausiliaria chiamata find_key la quale data una stringa ritorni un indice (numero naturale tra 0 e 25) tale che se s inizia per a o A, il valore di ritorno sarà 0; se s inizia con b o B, sarà 1, … se s inizia con z o Z sarà 25; in tutti gli altri casi solleva un'eccezione]

```
bool search_word(string s, dictionary d)

try {
   int ind=find_key (s);
   return is_in(s,D.d[ind]); //cerco nella lista degli elementi che
iniziano per s
}
catch … // catturo l'eccezione sollevata dalla find_key
```

NOTA: nel testo si è fatto spesso riferimento a funzioni che assumiamo avere… è utile usarle (ci si risparmia tanta fatica), in questo caso avevamo sia la find_key (che ritorna l'indice giusto della struttura dati) che la is_in (che cerca una parola in una back_of_word)

```
}
```

ESERCIZIO 4 [4 punti]

Scrivere una funzione *ricorsiva* che dato in input un array di elementi interi, ritorni il numero di elementi pari nell'array stesso

```
int count_even(int A[], unsigned int size) {
    if (size<=0) throw ERROR;
    if (size==1) return (A[size-1]%2==0); //base
    return (A[size-1]%2==0)+recoursive_count_even(A,size-1); //ricorsione
```

NOTE :
- Nella mia implementazione size rappresenta la lunghezza dell'array
- (A[size-1]%2==0) è un'espressione booleana (valore true/false); qui uso un cast implicito con int (valore 1 se l'espressione è true, ossia l'elemento che considero è pari, 0 altrimenti)

```
}
```