ESERCIZIO 1
Scrivere un programma che legga da terminale due sequenze di caratteri (di lunghezza compresa tra 1 e 8) da interpretare come cifre in base 7.
Ciascuna sequenza è conclusa da un carattere di spaziatura.

Il programma calcola la differenza fra il primo e il secondo valore e la visualizza a monitor come stringa in rappresentazione binaria in complemento a due.

Nel caso le stringhe non rispettino la specifica, il programma stamperà la stringa "errore 1" o "errore 2" sul terminale di errore (stderr) se ad essere stata acquisita in modo scorretto è stata la prima stringa oppure la seconda.
In entrambi i casi, il programma restituisce il valore 0 senza proseguire.
Tutte le stampe devono terminare con un carattere di “a capo”.

Esempi:

Input	**Output**
`4 7`	`errore 2`
`213 123`	`0101010`
`c`	`errore 1`

ESERCIZIO 2
Il biscarto è un gioco enigmistico che pone in relazione tre stringhe, ottenendo la terza dalla concatenazione delle prime due, da cui sia stata tolta una sottostringa comune.

Si progetti un programma di grado di determinare se tre stringhe lette da terminale costituiscono un biscarto.

In particolare, la funzione biscarto dovrà restituire -1 se la terna non costituisce un biscarto, e in caso contrario un numero intero positivo che rappresenti la posizione nella terza stringa della prima lettera derivante dalla seconda stringa.

Esempi:
pa**vo**ni / co**vo** = panico (scarto: "vo")
a**r**ma / to**r**re = amatore (scarto: "r")
casa / **ca**nto = santo (scarto: "ca")

Si consideri noto il seguente codice:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int biscarto(char* a, char* b, char* c);

int main() {
      char a[128], b[128], c[128];
      scanf("%s %s %s", a,b,c);
      printf("%d\n", biscarto(a,b,c));
      return 0;
} // end main
```

ESERCIZIO 3
I dati degli atleti di una società sportiva sono classificati in un file, che contiene per ogni riga la descrizione di un atleta, costituita dal suo nome (una stringa di caratteri alfabetici di al più 31 elementi), e da quattro numeri compresi tra 0 e 20, che rappresentano la valutazione della sua forza, agilità, velocità e resistenza.

Per esempio, un file di questo tipo potrebbe contenere i dati:

```
tizio             14    15    6     8
caio        10    12    10    12
sempronio   12    14    18    11
```

Per formare una squadra, è necessario che gli atleti siano dotati di caratteristiche compatibili fra loro. A tale scopo, si considera che gli atleti di una squadra siano compatibili se, per ciascuna delle loro caratteristiche, non vi è una distanza fra due atleti pari o superiore a 5.
Impiegando il main e i tipi di strutture dati forniti, realizzare un programma che:

- Carica da file i dati in una apposita struttura dati di tipo "squadra" (funzione "`carica_dati`");
- Verifica che una squadra sia compatibile, restituendo 1 in tale caso e 0 in caso contrario (funzione "`verifica`");

- Nel caso in cui vi siano più di sei atleti nella società, determina se sia possibile comporre una squadra di 5 elementi compatibili (funzione "`esiste_squadra`").

Codice fornito:

```
#include <stdio.h>
#include <stdlib.h>

typedef enum {
      FORZA, AGILITA,VELOCITA, RESISTENZA
} caratteristiche_t;

typedef struct {
      char nome[32];
      unsigned short caratteristiche[4];
      /* NOTA: Si possono usare i valori descritti da caratteristiche_t
      per indicizzare questo array */
} persona_t;

typedef struct {
      persona_t *elementi; /* Contiene i dati degli atleti */
      unsigned numero; /* Numero di atleti nel campo elementi */
} squadra_t;

squadra_t carica_dati(char* nome_file);

int verifica(squadra_t s);

int esiste_squadra(squadra_t s);

int main() {
      char nomef[100];
      scanf("%s", nomef);
      squadra_t s = carica_dati(nomef);
      printf("%d\n", s.numero);
      if (s.numero > 1 && s.numero < 5)
            printf("%d\n", verifica(s));
      if (s.numero > 5)
            printf("%d\n", esiste_squadra(s));
      return 0;
}
```

ESERCIZIO 4
Scrivere un programma C che acquisisca da terminale una stringa i cui caratteri sono da interpretare come cifre di un numero intero espresso in base dodici (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 'a'=10_{dec}, 'b'=11_{dec}).
La lunghezza della stringa può variare da 1 a 8.

Il programma deve calcolare il valore decimale del numero in base 12 memorizzato come stringa e riportare a video sia il numero minimo di bit necessari per rappresentarlo in binario naturale sia, sulla riga successiva, il valore del numero in formato esadecimale.
Nel caso la stringa non rispetti la specifica, il programma riporterà a video il numero -1.
In entrambi i casi, il programma restituisce il valore 0.

Esempi:
Input:
4
Output:
3
4

Input:
a3b
Output:
11
5cf

Input:
c
Output:
-1

ESERCIZIO 5
Si progetti una funzione `evolveMatrix(...)` che riceve in ingresso una matrice NxN contenente valori interi nell'insieme {0, 1}, e che modifica la matrice in modo tale che ogni cella della matrice contenga, dopo l'esecuzione della funzione, 1 se il conteggio dei valori non nulli nelle celle adiacenti (inclusa la cella corrente) è maggiore o uguale al conteggio dei valori nulli, 0 altrimenti.

Esempio con N == 4

Input:

1	0	0	1
0	1	0	1
1	0	1	0
1	1	0	1

Output:

1	0	1	1
1	0	0	1
1	1	1	1
1	1	1	1

ESERCIZIO 6

Scrivere un programma C che legga da terminale una stringa di lunghezza arbitraria, considerando soltanto i caratteri alfabetici (cioè quelli che rappresentano lettere maiuscole o minuscole, escluse quelle dotate di accenti o altri segni diacritici), e stampi a terminale:

1. Sulla prima riga, la mediana del numero di occorrenze dei caratteri alfabetici nella stringa letta (considerando identici i caratteri maiuscoli e minuscoli). La mediana in una sequenza numerica è definita come il "valore centrale" della sequenza ordinata in ordine ascendente. Se la sequenza è di lunghezza pari, la mediana è calcolata come la media aritmetica dei due valori centrali.
   Esempio: [1, 4, 5, 7], mediana: (4+5)/2= 4.5. [6, 11, 16], mediana: 11.
   Attenzione: per ogni esecuzione del programma la mediana deve essere calcolata considerando le lettere inserite (quindi quelle con num. di occorrenze >= 1) e non necessariamente tutto l'alfabeto.
2. Sulle righe successive, i caratteri (minuscoli, uno per riga) il cui numero di occorrenze è diverso da zero, seguiti dal segno ":" e dal proprio numero di occorrenze.
3. Sull'ultima riga, la sequenza di caratteri (minuscoli) il cui numero di occorrenze è maggiore della mediana.