ESERCIZIO 1
La moda statistica è, in un insieme di valori, quello che compare con la massima frequenza. È possibile che siano presenti più mode, ovvero diversi valori che compaiono tutti con la massima frequenza.

Progettare un programma che legge da terminale un numero, che rappresenta la lunghezza di una sequenza di caratteri alfabetici, e successivamente la sequenza stessa. Successivamente, il programma deve scrivere a terminale il valore della moda, seguita, una per riga, dall'elenco dei caratteri che compaiono con tale frequenza. I caratteri devono essere scritti in ordine alfabetico, e ognuno su una riga diversa.

Esempi:

Input	**Output**

`10`
`ababcdefba`
	`3`
	`a`
	`b`

Input	**Output**

`14`
`cbabcdefbacdbc`
	`4`
	`b`
	`c`

ESERCIZIO 2
Data in ingresso una matrice di numeri interi di dimensione definita dall'utente, calcolare se la matrice contiene due sottomatrici 2x2 la cui somma degli elementi sia uguale.

A questo scopo, implementare la funzione

```
int trova2x2sommapari(int *m, int r, int c);
```

dove `m` è la matrice, rappresentata come un array di interi, `r` è il numero di righe, `c` il numero di colonne, e restituisce `1` se tali matrici sono presenti (o `0` altrimenti).

Esempi:

Input	**Output**
3 3 1 2 3 4 1 3 0 9 1	1 2 3 4 1 3 0 9 1 1

Le sottomatrici:

4 1 0 9	1 3 9 1

hanno infatti entrambe somma degli elementi pari a 14.

Si consideri noto il seguente codice:

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

int *leggi_matrice(int r, int c){
    int *m=malloc(sizeof(int)*r*c);
    for(int i=0; i<r; i++)
        for(int j=0; j<c; j++)
            scanf("%d", &m[i*c+j]);
    return m;
}

void stampa_matrice(int *m, int r, int c){
    for(int i=0; i<r; i++) {
        for(int j=0; j<c; j++)
            printf("%d ", m[i*c+j]);
        printf("\n");
    }
}

int trova2x2sommapari(int *m, int r, int c);

int main(){
    int r, c;
    scanf("%d %d\n", &r, &c);
    int *m=leggi_matrice(r,c);
    stampa_matrice(m,r,c);
    printf("%d\n", trova2x2sommapari(m, r, c));
    free(m);
}
```

ESERCIZIO 3
Le netlist sono descrizioni di circuiti elettronici, rappresentati da liste di componenti e porte.
Un componente è definito dal proprio nome (un identificativo alfanumerico unico) e dall'operatore logico che implementa (and, or, not).
Una porta è definita dal proprio nome, e dai nomi di due componenti che la porta collega fra loro.
Tali definizioni sono salvate su file, una per riga.

Esempio:
```
componente a and
componente b or
porta e a b
componente c or
porta f b c
porta g a c
```

Scrivere un programma che legga da file una netlist e verifichi se, per ciascuna porta, è rispettata la seguente proprietà: i componenti che la porta connette devono essere dichiarati nel file, nelle righe precedenti a quella in cui è dichiarata la porta stessa.
Il programma stampa, in primo luogo, il numero e la lista dei nomi dei componenti delle porte letti, come indicato nel codice allegato.
Il programma stampa poi a terminale, uno per riga, i nomi delle porte dichiarate in modo erroneo.

I test sono separati in tre categorie:
1. [6 punti] File che non presentano errori (un file vuoto è considerato corretto, e deve produrre una struttura dati vuota).
2. [2 punti] File che presentano errori dovuti al fatto che le porte sono dichiarate usando componenti non definiti nel file, oppure usando i nomi di altre porte in luogo di quello di un componente.
3. [2 punti] File che presentano, oltre eventualmente agli errori di prima, anche porte dichiarate usando componenti presenti nel file, ma definiti in righe successive a quella in cui è definita la porta.

Codice fornito (e da non modificare):

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

typedef struct {
    char nome[32];   // nome della porta
    char input[32];  // componente 1, input
    char output[32]; // componente 2, output
    int numero_riga; // numero di linea nel file
} t_porta;

typedef struct {
    char nome[32];      // nome del componente
    char operatore[32]; // operatore
    int numero_riga;    // numero di linea nel file
} t_componente;

typedef struct {
    t_componente *componenti; // array dinamico di componenti
    t_porta *porte;           // array dinamico di porte
    int np;                   // lunghezza dell'array delle porte
    int nc;                   // lunghezza dell'array dei componenti
} t_circuito;

// Dichiarazioni anticipate
t_circuito carica_dati(char *nomefile);

void stampa_errori(t_circuito c);

// Funzioni di supporto
```

```
void stampa_circuito(t_circuito c){
    if (c.nc != 0) {
        for (int i = 0; i < c.nc; i++)
            printf("%s ", c.componenti[i].nome);
        printf("\n");
    }

    if (c.np != 0) {
        for (int i = 0; i < c.np; i++)
            printf("%s ", c.porte[i].nome);
        printf("\n");
    }
}

int main(){
    char nomef[128];
    scanf("%s", nomef);
    t_circuito c = carica_dati(nomef);
    printf("componenti = %d\n", c.nc);
    printf("porte = %d\n", c.np);
    printf("circuito:\n");
    stampa_circuito(c);
    printf("errori:\n");
    stampa_errori(c);
}
```

ESERCIZIO 4

Progettare e implementare la funzione `int base12()` che legga da terminale una sequenza di caratteri, e ne interpreti i primi 8 come un numero intero in base 12, con eventuale segno iniziale.

La sequenza può essere più corta di 8 caratteri, e in tal caso è terminata da un carattere di spazio. Se la sequenza non è interpretabile come un numero in base 12, la funzione deve restituire il più grande numero negativo non rappresentabile con tale specifica (nota: tale numero è indipendente dall'input).

Esempi:

	Input	**Output**
•	a	10
•	-a	-10
•	000000011	1

Si consideri il seguente codice come noto:

```
#include <stdio.h>

int base12();

int main() {
      printf("%d\n", base12());
}
```

ESERCIZIO 5
Il metagramma è un gioco enigmistico che consiste nel passaggio da una parola ad un'altra, di uguale lunghezza, attraverso passi intermedi in cui ogni nuova parola è generata cambiando una sola lettera alla volta.

Esempio:
GATTO
GETTO
PETTO
PESTO
PESCO
PESCE

Scrivere un sottoprogramma C di prototipo `int metagramma(char m[][N], int righe)` che acquisisca come parametro una matrice di caratteri e il numero di righe della matrice e verifichi se le sue righe contengono dall'alto verso il basso un metagramma.
La funzione restituisce il valore di righe se la matrice contiene un metagramma, o l'indice della riga in cui si trova la prima parola che non rispetta la regola del metagramma (la prima parola è sempre valida, e la prima riga ha indice 0).

Nota: le parole contenute in ogni riga della matrice possono essere composte da un numero di caratteri inferiore rispetto al numero di colonne. In tal caso la parola termina con un '\0'.

Si consideri il seguente codice come noto:

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

#define N 8

int metagramma(char m[][N], int l);

int main() {
	int n=0, i;
	char s[N+1];
	scanf("%d", &n);
	char matrice[n][N];

	for(i = 0; i<n; i++) {
		scanf("%s", s);
		strncpy(matrice[i],s,N);
	}

	printf("%d\n",metagramma(matrice,n));

	return 0;
}
```

ESERCIZIO 6
Sia una whishlist definita in un file in modo che per ogni riga ci sia il nome dell'oggetto e il suo prezzo in euro, separato da uno spazio.

Esempio:
```
Televisione 500
Smartphone 200
Cuffie 50
Aspirapolvere 400
Computer 1200
Videogame 70
```

Scrivere un programma che legga una whishlist da un file che, dopo aver letto da standard input un budget in euro, stampi in output, uno per riga, il numero di riga degli oggetti che si possono comprare in un solo acquisto con quel budget. Nota che il programma deve stampare gli oggetti che facciano spendere il più possibile, sempre restando all'interno del budget.

Per semplicità, non si consideri il caso con soluzioni multiple.

Si consideri il seguente codice come noto:

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

typedef struct {
    char nome[32];          // nome del prodotto
    int prezzo;             // prezzo del prodotto
} t_prodotto;

typedef struct {
    t_prodotto *prodotti;   // lista di prodotti
    int np;                 // numero prodotti
} t_whishlist;

t_whishlist carica_dati(char* nomefile);

void stampa_oggetti_acquistabili(t_whishlist whishlist, int budget);

void stampa_whishlist(t_whishlist whishlist) {
    printf("Ecco la whishlist:\n");
    for (int i=0; i<whishlist.np; i++) {
        printf("%d: %s %d\n", i, whishlist.prodotti[i].nome,
                              whishlist.prodotti[i].prezzo);
    }
}

int main() {
    t_whishlist whishlist;
    int budget;
    char nomefile[128];

    scanf("%s", nomefile);
    scanf("%d", &budget);

    whishlist = carica_dati(nomefile);
    stampa_whishlist(whishlist);
    stampa_oggetti_acquistabili(whishlist, budget);
}
```