



Cognome	Nome	Matricola	Voto: .../30
---------	------	-----------	--------------

Istruzioni:

- questa prova concorre al voto finale (insieme al voto della prima prova in itinere e al voto di laboratorio) solo se ha una valutazione non inferiore a 18/30;
- non è possibile consultare libri, appunti, né comunicare;
- non è possibile utilizzare la calcolatrice o qualsiasi dispositivo elettronico;
- si può rispondere ai quesiti nell'ordine preferito;
- si può scrivere con qualsiasi colore, anche a matita, ad eccezione del **rosso**.

Stile del codice C:

- non è necessario inserire direttive `#include`.
- i commenti non sono necessari, ma potrebbero essere utili nel caso di errore per capire le intenzioni.
- è possibile (e consigliato) utilizzare funzioni di libreria.

Quesito 1 (5 punti)

Punteggio ottenuto: .../5

Scrivere un programma che chiede all'utente il nome del file sorgente, il nome del file destinazione (entrambi di al più 40 caratteri) ed un numero intero n . Il programma copia nel file destinazione i primi n caratteri presenti nel file sorgente (dovessero essercene meno, ne copia meno). Gestire i casi di errore nell'accesso ai file.

Quesito 2 (6 punti)

Punteggio ottenuto: .../6

Scrivere un sottoprogramma che ricevuto in ingresso un array monodimensionale e qualsiasi parametro ritenuto strettamente necessario calcola e *trasmette* al chiamante l'elemento della sottosequenza di elementi adiacenti uguali tra loro più lunga e la sua lunghezza.

Esempio: **Ingresso:** 128337466663228888833
 Uscita: elemento: 8
 lunghezza sottosequenza: 5

Quesito 3 (6 punti)

Punteggio ottenuto: .../6

Scrivere un sottoprogramma che ricevuta in ingresso una testa di lista (per la gestione di stringhe di lunghezza a priori ignota) ed un numero n , elimina dalla lista tutti gli elementi contenenti una stringa più lunga di n caratteri, o che contiene il carattere `-`. Si organizzi il sottoprogramma in modo che si avvalga di altri sottoprogrammi; inoltre si consideri dato il sottoprogramma per eliminare un elemento da una lista in base alla stringa in esso contenuta, il cui prototipo è: `lista_t * deleteElem(lista_t *, char *)`;

Esempio: **Ingresso:** casa → indescrivibile → equo → self-service → archivio
 n: 13
 Uscita: casa → equo → archivio

Quesito 4 (6 punti)

Punteggio ottenuto: .../6

Scrivere un programma che chiede all'utente quanti processi numerati interi verranno inseriti poi dall'utente e quindi li acquisisce tutti. Al termine dell'acquisizione il programma visualizza la scritta "dati in ordine" se i dati inseriti sono ordinati in modo crescente, "dati in ordine qualsiasi" in caso contrario.

Quesito 5 (7 punti)

Punteggio ottenuto: .../7

Si consideri il gioco "Alberi", costituito da una griglia quadrata, in cui in ogni elemento può esserci un albero o meno. Le dimensioni della griglia variano da gioco a gioco, ma sono di al più 10 elementi per lato. La griglia è occupata da aree di colore diverso, cui appartengono uno o più elementi, e le regole per la corretta presenza degli alberi sono le seguenti: i) ci devono essere 2 alberi per riga, ii) ci devono essere 2 alberi per colonna, iii) ci devono essere 2 alberi per colore, iv) negli elementi adiacenti ad un albero non ci devono essere alberi. Gli schemi più piccoli contengono 1 solo albero per riga, colonna e colore.

1) Si deve realizzare un sottoprogramma `erroreGriglia()` che utilizzando sottoprogrammi già esistenti, controlli la correttezza di uno schema ricevuto in ingresso (oltre a qualsiasi altra informazione ritenuta strettamente necessaria) e restituisca 0 se è valida, 1 altrimenti.

2) Si scriva il codice del sottoprogramma `checkColore()` seguendo il prototipo indicato.

Si considerino sottoprogrammi già esistenti quelli di seguito indicati:

```
int checkRiga(gridElem * rowGrid, int dim, int nAlberi);
    restituisce 0 se il num. di alberi nella riga rowGrid, di dimensione dim è quello richiesto (nAlberi);
int checkColonna(gridElem Grid[][MAXDIM], int dim, int col, int nAlberi);
    restituisce 0 se il num. di alberi in colonna col della griglia Grid, di dimensione dim, è quello
    richiesto (nAlberi);
int checkColore(gridElem Grid[][MAXDIM], int dim, int color, int nAlberi);
    restituisce 0 se il num. di alberi sull'area di colore color della griglia Grid, di dimensione dim,
    è quello richiesto (nAlberi);
int checkDistanza(gridElem Grid[][MAXDIM], int dim);
    restituisce 0 se quando si incontra un albero nella griglia Grid, di dimensione dim, esso non ha alberi negli
    elementi adiacenti.
```

Si riporta qui il main (e tutte le strutture dati esistenti).

```
#define MAXDIM 15

typedef struct _gridElem{
    int albero; /* 1: si albero 0: vuoto*/
    int colore; /* id del colore*/
} gridElem;

typedef struct _listColor{
    int colore;
    struct _listColor * next;
} listColor;

int main(int argc, char * argv[])
{
    int dimGrid; /* dimensione dell'attuale griglia */
    int numAlberi; /* numero alberi per riga/colonna/colore */
    char * NomeFile; /* nome del file con griglia*/
    listColor * ElencoColori = NULL; /* lista dei colori diversi */
    gridElem Griglia[MAXDIM][MAXDIM]; /* griglia */

    if(argc < 2){
        printf("alberi filegriglia\n");
        return -1;
    }
    NomeFile = argv[1];
    if(LeggiGriglia(NomeFile, Griglia, MAXDIM, &dimGrid, &numAlberi, &ElencoColori)){
        stampaGriglia(Griglia, dimGrid);
        if(!erroreGriglia(...))
            printf("Congratulazioni: griglia corrispondente ad una configurazione valida\n");
        else
            printf("Griglia non corrispondente ad una configurazione valida\n");
    } else {
        printf("\nerrore: impossibile leggere una griglia valida dal file %s\n", NomeFile);
        return -1;
    }
    return 0;
}
```



