Contesto

Il G.I.S. (*Geographical Information System*), o sistema informativo geografico, è uno strumento informatico che permette di rappresentare, analizzare, e interrogare entità o eventi che si verificano sul territorio.
In questo contesto, un modello digitale di elevazione rappresenta la mappa di un territorio attraverso una matrice rettangolare con `NUMR` × `NUMC` celle, in cui ogni cella memorizza un valore in virgola mobile corrispondente alla misura, in centimetri, della quota sul livello del mare rilevata nell'omologo punto sul territorio.

Si considerino dunque i seguenti tipi di dato, utili per rappresentare in `C` la mappa di un territorio.

```
#define NUMR  50
#define NUMC  80

typedef float mappa[NUMR][NUMC];
typedef struct { int r, c; } posizione;
```

La mappa, `m`, di un qualunque territorio è assunto che abbia un numero di righe minore o uguale a `NUMR` e un numero di colonne minore o uguale a `NUMC`.
Un **percorso carrabile valido**, `pcv`, è definito come un vettore di tipo `posizione`, in cui celle consecutive memorizzano posizioni a cui corrispondono celle contigue di `m` (in orizzontale o in diagonale) e aventi differenze di quota minore o uguale a 15 cm.

Nota:
Per semplicità, su una mappa si considerano solo percorsi che vanno da sinistra a destra, cioé dalla colonna `0` alla colonna (`NUMC-1`), senza cicli e senza tratti verticali.
Dunque, nella creazione di un percorso, essendo possibili solo movimenti in avanti, se si parte dalla posizione `[riga][colonna]` ci si può spostare solo in una cella di posizione

- `[riga-1][colonna+1]` (solo se la riga non è la prima)
- `[riga][colonna+1]`
- `[riga+1][colonna+1]` (solo se la riga non è l'ultima)

Esercizio 27 (8 punti)

Si scrivano in C i due sottoprogrammi seguenti.

- Considerando come parametri in ingresso una mappa `m` e un vettore di posizioni, il sottoprogramma

  `...valido(...)`

  deve ritornare al chiamante il valore intero `1`, se il vettore di posizioni rappresenta un percorso carrabile valido sulla mappa; in caso contrario deve ritornare `0`.

- Considerando come parametri in ingresso una mappa `m` e un vettore di posizioni rappresentante un percorso valido, il sottoprogramma

  `...stampaPercorso(...)`

  deve visualizzare a video la mappa del territorio, riportando ogni cella lungo il percorso con un carattere `'*'` e ogni altra cella con un carattere `'.'`.

Soluzione:

```
float absDiff(float v1, float v2) {
    if (v1 < v2) return v2-v1;
    else return v1-v2;
}

int valido(mappa m, posizione p[NUMC]) {
   for (int j = 1; j < NUMC; j++) {
      if ( p[j].c != 1 + p[j-1].c          ||
           absDiff(p[j].r, p[j-1].r) > 1   ||
           absDiff(m[p[j].r][p[j].c],
                   m[p[j-1].r][p[j-1].c]) > 15
         )
        return 0;
   }
   return 1;
}

void stampaPercorso(mappa m, posizione p[NUMC]) {
    if ( valido(mappa, p) == 0 ) {
      printf("\n Il percorso indicato non e' valido ! \n");
      return;
    }
    for (int i = 0; i < NUMR; i++) {
      for (int j = 0; j < NUMC; j++) {
         if (i == p[j].r && j == p[j].c)
           printf("*");
         else
           printf(".");
      }
      printf("\n");
    }
}
```

Esercizio 2 (10 punti)

Si assuma di avere a disposizione un *file* di caratteri che contiene la mappa di un territorio.
Il *file* contiene valori di quota organizzati in esattamente `NUMR` righe e `NUMC` colonne, più precisamente:

- la prima riga contiene `NUMC` valori di quota da memorizzare nella prima riga della matrice; ciascun valore è separato dal successivo da uno spazio (`' '`), tranne l'ultimo valore che è invece seguito da un fine-riga (`'\n'`);
- la seconda riga contiene i valori di quota da memorizzare nella seconda riga della matrice ; ciascun valore è separato dal successivo da uno spazio (`' '`), tranne l'ultimo valore che è invece seguito da un fine-riga (`'\n'`); ecc.

Si assuma inoltre di avere a disposizione il sottoprogramma

```
int cercaPercorso(mappa m, int riga, int col, posizione cammino[NUMC]);
```

la cui invocazione consente di cercare un percorso carrabile valido a partire dalla posizione `[riga][col]` sulla mappa `m`, e memorizzare nell'*array* `cammino` le posizioni che lo compongono. Il sottoprogramma ritorna `1` se un attraversamento valido viene trovato, `0` altrimenti.

Scrivere un programma principale, `main(){ ... }`, che chieda all'utente il nome del *file* contenete la mappa di un territorio, lo apra e carichi in una variabile `m`, di tipo mappa, la matrice in esso riportata. Il programma deve inoltre riportare a video i percorsi validi che vanno dalla colonna `0` alla colonna `NUMC-1` sulla mappa.

Nota: fare uso del sottoprogramma `...cercaPercorso(...)` assunto a disposizione per quest'esercizio e del sottoprogramma `...stampaPercorso(...)`, la cui funzionalità è stata descritta nell'esercizio precedente.

Soluzione:

```
#include <stdio.h>   // per utilizzare: scanf, printf, fscanf,
                     // fprintf, fopen, ferror, fclose
#include <stdlib.h>  // per utilizzare: exit

#define NUMR  50
#define NUMC  80
typedef float mappa[NUMR][NUMC];
typedef struct { int r, c; } posizione;

int cercaPercorso(mappa, int, int, posizione cammino[NUMC]);
int valido(mappa, posizione p[NUMC]);
void stampaPercorso(mappa, posizione p[NUMC]);
```

```
int main() {

   char nomeFile[30+1] = { '\0' };
```

```
    printf("\n Nome del file con la mappa(max 30car): ", nomeFile);
    scanf("%31s", nomeFile);

    FILE* fp = fopen(nomeFile, "r");
    if ( fp == NULL ) {
      fprintf(stderr, "\n Il file %s non esite o e' corrotto! ");
      exit(-1);
    }

    mappa m;

    for (int i = 0; i < NUMR; ++i) {
       for (int j = 0; j < NUMC; ++j) {
          fscanf(fp, "%f", &m[i][j]);
          if ( ferror(fp) != 0 ) {
            fprintf(stderr, "\n Errore nella lettura da file! ");
            exit(-1);
          } // end if
       } // end for j
    } // end for i

    fclose(fp);

    int trovatoValido = 0, almenoUno = 0;
    posizione cammino[NUMC] = { 0 };

    for (int i = 0; i < NUMR; i++) {
       trovatoValido = cercaPercorso(m, i, 0, cammino);
       if ( trovatoValido == 1 ) {
         almenoUno = 1;
         printf("\n Percorso con inizio da cella [%d][%d]\n", i, 0);
         stampaPercorso(m, cammino);
       } // end if
    } // end for i

    if ( almenoUno == 0 )
      printf("\n Non ci sono percorsi carrabili validi !\n");

    return 0;

} // end main


// ... qui la definizione dei sottoprogrammi utilizzati ...
```

Esercizio 3 (6 punti)

Scrivere un sottoprogramma `C` che prendendo come parametri una mappa `m` e un vettore di posizioni, restituisca una lista dinamica semplicemente concatenata non vuota, se il vettore contiene un percorso valido. I nodi della lista dinamica devono memorizzare le posizioni presenti nel percorso nello stesso ordine in cui compaiono nel vettore di posizioni.

Definire anche le strutture dati opportune, tenendo conto che ogni nodo della lista deve memorizzare una posizione lungo il percorso e il valore di quota a esso corrispondente sulla mappa `m`.

Soluzione:

```
typedef struct nodo_t { posizione p;
                        float quota;
                        struct nodo_t* next;
        } nodo;

nodo* creaLista(mappa m, posizione cammino[NUMC]) {

     nodo *testa = NULL, *nuovo, *ultimo;

     if (valido(m, cammino) == 0) return testa;

     for (int y = 0; y < NUMC; y++) {
        nuovo = (nodo*) malloc(sizeof(nodo));
        nuovo->p = cammino[y];
        nuovo->quota = m[(nuovo->p).r][(nuovo->p).c];
        nuovo->next = NULL;

        if (testa == NULL) {
          testa = ultimo = nuovo;
        }
        else {
          ultimo->next = nuovo;
        }

     return testa;
}
```