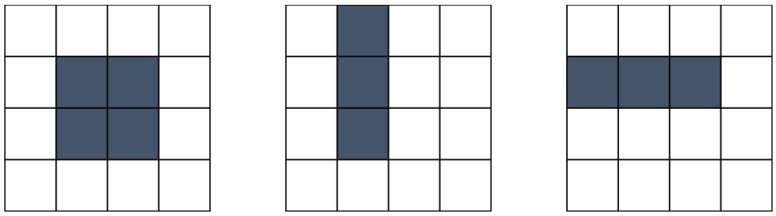
	Politecnico di Milano Scuola di Ingegneria Industriale e dell'Informazione **FONDAMENTI DI INFORMATICA** Appello 2 Luglio 2018		COGNOME E NOME
	RIGA	COLONNA	CODICE PERSONA
			Spazio riservato ai docenti

- Il presente plico contiene 4 esercizi e **deve essere debitamente compilato con cognome e nome, codice persona**.
- Il tempo a disposizione è di 1 ora e 45 minuti.
- Non separate questi fogli. Scrivete la soluzione solo sui fogli distribuiti, utilizzando il retro delle pagine in caso di necessità. Cancellate le parti di brutta con un tratto di penna.
- Ogni parte non cancellata a penna sarà considerata parte integrante della soluzione.
- È possibile scrivere a matita (e non occorre ricalcare al momento della consegna!).
- **È vietato utilizzare telefoni o pc**. Chi tenti di farlo vedrà annullata la sua prova.
- **Non è ammessa la consultazione di libri e appunti.**
- **Qualsiasi tentativo di comunicare con altri studenti comporta l'espulsione dall'aula**.
- È possibile ritirarsi senza penalità.
- **Non è possibile lasciare l'aula conservando il tema della prova in corso**.

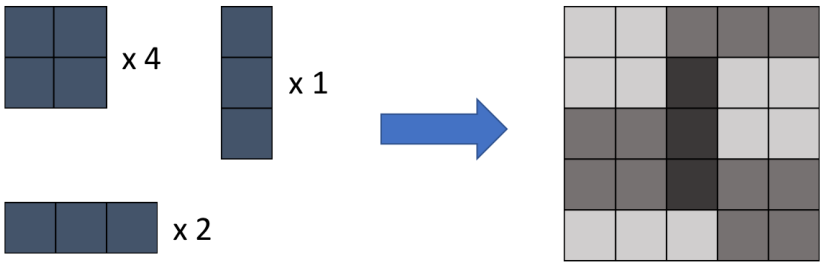
Esercizio 1 (12 punti)

INCASTRI è un semplice rompicapo con le seguenti regole. Per risolvere il rompicapo, è necessario posizionare il set di pezzi fornito su una griglia di dimensione 5x5. Ogni pezzo può essere posizionato ovunque, ma **(i) non è possibile ruotarlo e (ii) non può sovrapporsi (neanche parzialmente) ad altri pezzi.**

I pezzi del gioco sono di tre diverse forme (quadrato, barra verticale e barra orizzontale) come mostrato dalla seguente figura (in cui è anche riportato un numero che identifica ciascuno dei tre diversi pezzi del gioco):



Il rompicapo viene risolto se tutti i pezzi vengono posizionati sulla griglia, come nel seguente esempio (attenzione, perché la soluzione sia valida **NON** è necessario che la griglia sia completamente piena):



Si assuma di rappresentare la griglia 5x5 nella memoria del calcolatore come una matrice di interi, in cui il valore 0 è usato per rappresentare una cella vuota e il valore 1 per rappresentare una cella occupata; inoltre si assuma di rappresentare con il valore 1, 2 e 3 rispettivamente i pezzi di forma quadrata, barra verticale e barra orizzontale.

Implementare in C le seguenti due funzioni:

A) `int fit(int G[5][5], int p, int i, int j)`
che restituisce 1 se è possibile posizionare nella cella <**i,j**> della griglia **G**, un pezzo la cui forma è definita dal parametro di ingresso **p**, viceversa restituisce 0.

Note. La matrice **G** può avere delle celle occupate (cioè non reppresenta necessariamente una griglia completamente vuota). Per posizionare un pezzo nella cella **<i,j>**, si intende posizionare il pezzo nella griglia in modo che la sua cella in alto a sinistra si trovi nella posizione **<i,j>** della griglia.

B) `int solve(int G[5][5], int p[], int npezzi)`
che restituisce 1 se è possibile posizionare tutti i pezzi specificati nel vettore di ingresso **p** di dimensione **npezzi**, all'interno della griglia **G**. Viceversa, la funzione restituisce 0.
Nell'implementare questa funzione è possibile utilizzare sia la funzione implementata al punto A sia le seguenti due funzioni (che non è necessario implementare):
- `int addp(int G[5][5], int p, int i, int j)` che aggiunge un pezzo di forma **p** alla griglia **G** in posizione **<i,j>** (occupando le opportune celle). La funzione restituisce 1 se ha successo, 0 altrimenti.
- `int removep(int G[5,5], int p, int i, int j)` che rimuove un pezzo di forma **p** alla griglia **G** in posizione **<i,j>** (liberando le opportune celle). La funzione restituisce 1 se ha successo, 0 altrimenti.

Note. Il problema può essere risolto ricorsivamente: è possibile posizionare sulla griglia **G**, i pezzi contenuti nel vettore **p**, se esiste almeno una posizione **<i,j>** di **G** dove può essere posizionato il primo pezzo di **p** ed è possibile posizionare tutti i rimanenti pezzi in **p** nella griglia **G** (in cui è stato aggiunto il primo pezzo in posizione **<i,j>**).

Soluzione

A.

```
int fit(int G[5][5], int p, int i, int j)
{

  if  (i<0 || j<0 || i>4 || j>4)
    return 0;
  if (p==1)
    if (i<4 && j<4)
      return (G[i][j]==0 && G[i+1][j]==0 && G[i][j+1]==0 && G[i+1][j+1]==0);
  if (p==2)
    if (i<3)
      return (G[i][j]==0 && G[i+1][j]==0 && G[i+2][j]==0);
  if (p==3)
    if (j<3)
      return (G[i][j]==0 && G[i][j+1]==0 && G[i][j+2]==0);

  return 0;
}
```

B.

```
int solve(int G[5][5], int p[], int npezzi)
{
  if (npezzi==0)
    return 1;
  else
  {
    int i,j;
    for (i=0; i<5; i++)
      for (j=0; j<5; j++)
        if (addp(G,p[0],i,j))
        {
          if (solve(G,p+1,npezzi-1))
          {
            removep(G,p[0],i,j);
            return 1;
          }
          else
            removep(G,p[0],i,j);
        }
     return 0;
  }
}
```

Esercizio 2 (8 punti)

Si consideri la seguente struttura dati per rappresentare una lista di numeri interi:

```
struct nodo
{
  int data;
  struct nodo *next;
};

typedef struct nodo *lista;
```

Si implementi la funzione `lista interseca(lista a, lista b)`, che riceve in ingresso la testa di due liste (a e b) e restituisce la testa di una lista che contiene tutti gli elementi che sono presenti sia nella lista a che nella lista b.

Note. Gli elementi nella lista restituita possono comparire in qualsiasi ordine. È possibile richiamare una funzione per l'inserimento di un elemento (in coda o in testa), ma è necessario fornirne l'implementazione.

Soluzione

```
lista interseca(lista a, lista b)
{
  lista c = NULL, cur, temp;
  int found;

  while (b != NULL)
  {
    found = 0;
    cur = a;
    while (cur && cur->data != b->data)
      cur = cur->next;
    if (cur != NULL) // se non è NULL, è stato trovato nella lista A
    {
      temp = malloc(sizeof(struct nodo));
      temp->data = b->data;
      temp->next = c;
      c = temp;
    }
    b = b->next;
  }
  return c;
}
```

Esercizio 3 (8 punti)

La società E Corp. è in possesso di una grande quantità di messaggi twitter su cui vuole condurre indagini di mercato. Per ogni messaggio twitter, devono essere memorizzate le seguenti informazioni: testo del messaggio (non più di 280 caratteri), lista degli hashtag contenuti nel messaggio (ogni hashtag è una sequenza di caratteri senza spazi e di lunghezza non superiore a 279 caratteri, che l'autore del messaggio indica come parola chiave), numero di hastag contenuti nel messaggio (che sono sicuramente inferiori a 100), account twitter dell'autore del messaggio (non più di 50 caratteri e senza spazi).

A. Creare il tipo di dato C, `tweet`, in grado di contenere tutte le informazioni rilevanti di un messaggio twitter sopra elencate.
B. Implementare la funzione, `float analizza(tweet collezione[], int n, char hashtag[])` che riceve in ingresso una **collezione** di **n** messaggi twitter ed un **hashtag**. La funzione restituisce la frazione di messaggi twitter contenuti nel parametro in ingresso collezione che (i) contengono l'hashtag passato come parametro in ingresso e (ii) hanno una lunghezza non superiore a 140 caratteri.

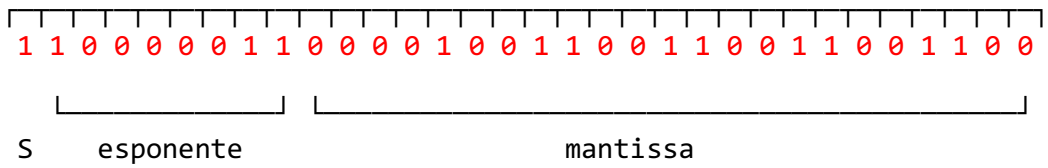
Soluzione

```
typedef char htag[280];

typedef struct
{
  char testo[281];
  htag htags[100];
  int nhtags;
  char account[51];
} tweet;

float analizza(tweet collezione[], int n, char hashtag[])
{
  int i,j;
  float r=0;
  for (i=0;i<n;i++)
  {
    if (strlen(collezione[i].testo)<=140)
      for (j=0; j<collezione[i].nhtags; j++)
        if (strcmp(collezione[i].htags[j],hashtag)==0)
        {
          r++;
          break;
        }
  }
  return r/n;
}
```

Esercizio 4 (4 punti)

Si consideri il numero negativo e razionale -16.6; riportare nello spazio seguente la sua codifica secondo lo standard IEEE a precisione singola (riportare di seguito anche i calcoli effettuati). Dire inoltre se la codifica è esatta (giustificare la risposta).

Soluzione



S:1

$16_{10} = 10000_2$

.6 * 2 = 1.2
.2 *2 = 0.4
.4 *2 = 0.8
.8 *2 = 1.6
…

$0.6_{10} = 0.\overline{1001}_2$

10000.10011001100110011001100 → $1.00001001100110011001100 * 2^4$

M: 00001001100110011001100
E = 127+4 = 131 → E:10000011

La codifica non è esatta ma approssimata, dal momento che la codifica della parte frazionaria è periodica.