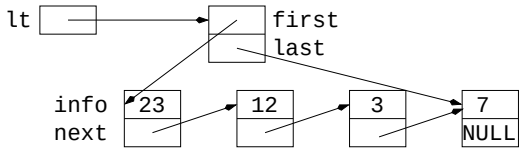
# Laboratorio di Linguaggi di Algoritmi e Strutture Dati
# Prova di laboratorio, turno 2, 18 giugno 2002

Si consideri il tipo `list` delle liste di interi come definito nel file `lib.h`:

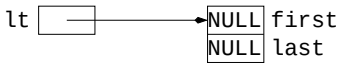
```
typedef struct lcell *list;

typedef struct cell *link;

struct lcell{
  link first,last;
};

struct cell{
  int info;
  link next;
};
```

I campi `first` e `last` contengono rispettivamente il puntatore alla prima e all'ultima cella della lista.

Ad esempio, se la variabile `lt` contiene la lista formata dalla successione `23 12 3 7`, allora la configurazione della memoria sarà:




La lista vuota è rappresentata da un indirizzo NON NULLO che punta alla struttura `struct lcell` dove i due campi `first` e `last` contengono il puntatore NULLO:




1. **(punti 0)** Si definisca la funzione `void rev_print(list l)` che stampa su video il contenuto di `l` in ordine inverso, ossia partendo dalla coda.

2. **(punti 1)** Si definisca la funzione `list reverse(list l1)` che restituisce una lista ottenuta invertendo l'ordine degli elementi di `l1`.

   Utilizzare una definizione ricorsiva basata sulla seguente proprietà: se una lista $l$ viene spezzata a metà in due liste $l_1$ e $l_2$ (ossia $l_1$ concatenata con $l_2$ è uguale a $l$), allora

   $$reverse(l) = cat(reverse(l_2), reverse(l_1))$$

   doce *cat* è la concatenazione di liste.

   **Nota bene:** qualsiasi altro tipo di algoritmo, anche se corretto, verrà valutato 0 punti!

   La funzione deve essere definita usando le seguenti due funzioni di `lib.o`:

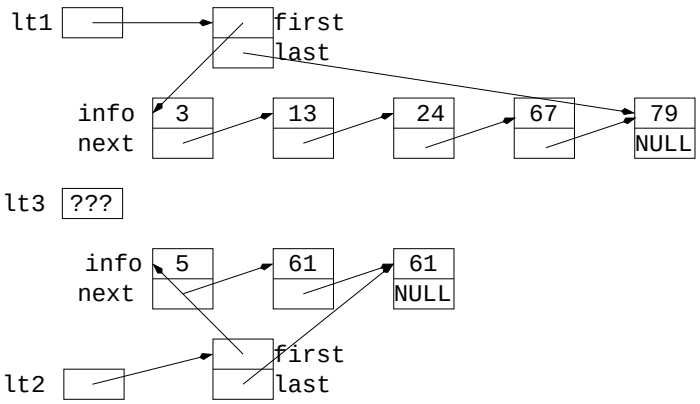
   - `list cat(list l1,list l2)` restituisce una nuova lista ottenuta concatenando `l1` davanti a `l2`. Per ulteriori spiegazioni si veda il punto 4.

- `list split(list l1)` restituisce una lista contenente la seconda metà di `l1` e lascia in `l1` solo la prima metà; nel caso `l1` abbia un numero dispari di elementi, l'elemento in eccesso apparterrà alla prima metà. Per ulteriori spiegazioni si veda il punto 4.

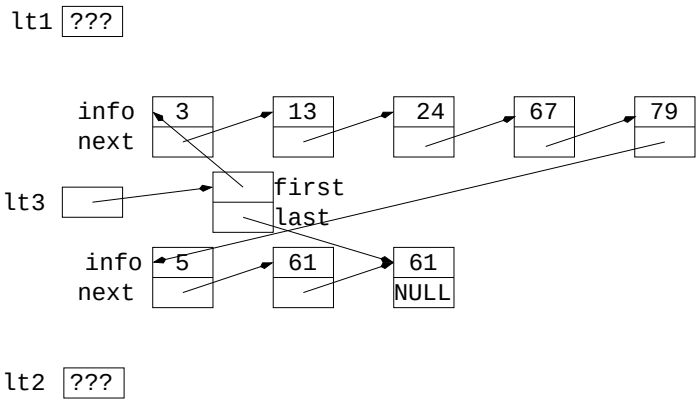
3. **(punti 2)** Si definisca la funzione `list cat(list l1,list l2)` utilizzata al punto 2. Come già specificato, la funzione restituisce una nuova lista ottenuta concatenando `l1` davanti a `l2`.

   La lista restituita deve puntare a una nuova cella di tipo `struct lcell`, mentre le celle di tipo `struct lcell` corrispondenti a `l1` e `l2` devono essere deallocate. Al contrario, la funzione non deve creare né distruggere celle di tipo `struct cell`.

   A titolo di esempio, si considerino le variabili `lt1` e `lt2` contenenti le seguenti liste:

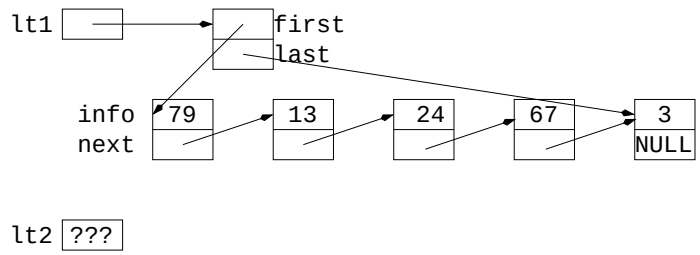



   Dopo l'istruzione `lt3=cat(lt1,lt2)`, si ottiene la seguente configurazione (dove `???` significa valore indefinito):




4. **(punti 2)** Si definisca la funzione `list split(list l1)` utilizzata al punto 2. Come già specificato, la funzione restituisce una nuova lista contenente la seconda metà di `l1` e lascia in `l1` solo la prima metà; nel caso `l1` abbia un numero dispari di elementi, l'elemento in eccesso apparterrà alla prima metà. La funzione non deve creare né distruggere celle di tipo `struct cell`.

   A titolo di esempio, si consideri la variabile `lt1` contenente la seguente lista:



Dopo l'istruzione `lt2=split(lt1)`, si ottiene la seguente configurazione: