```
Esercizio 1.
Si scriva una funzione C ricorsiva che, preso in input un numero intero,
ne stampa le cifre decimali in ordine inverso, ovvero dalla meno significativa alla
più significativa.
La vostra  soluzione è tail recursive? Si motivi la riposta.
Esempio. Se il numero intero ricevuto come parametro di input è 1234, la funzione dev
e stampare 4321. */


Esercizio  2
Si scriva una funzione ricorsiva in coda che riceve in input una lista concatenata  d
i stringhe e restituisce
una lista di stringhe di circa metà elementi in cui l'i-simo (i≤1) elemento contiene
la concatenazione
degli elementi in posizione 2i-1 e 2i (se esiste)  della lista originaria, che deve
rimanere invariata.

Esempio:

input : ab -> cd -> ef
Output : abcd -> ef


Esercizio 3.

Si scriva una funzione che, mediante scambi della posizione dei figli dei nodi di un
albero binario
(sottoalbero destro che diventa sottoalbero sinistro e viceversa), riorganizza un
albero in modo tale che, per ogni nodo,
 il sottoalbero sinistro contiene un numero di nodi maggiore o uguale di quello dei
nodi del sottoalbero destro.
  Analizzare la complessità computazionale della soluzione proposta.
Nota. E' possibile fornire una soluzione che risolve l'esercizio in tempo lineare.


Esercizio 4
Si individuino opportuni dati di test per la seguente funzione,
evidenziando per quali dati si e' seguito l'approccio a scatola nera e per quali
quello a scatola trasparente.
Si dia una breve motivazione della scelta.


char * Rstringchr (char *s, int c)
/*individua la prima occorrenza del carattere c nella stringa s*/
/*Pre: s!= NULL*/
/* Post: restituisce il puntatore alla prima occorrenza di c nella stringa. Se tale
occorrenza non esiste, restituisce NULL*/

{if (*s =='\0')
        return (NULL);
    else if (*s == c)
        return s;
    else
```

```
        return (Rstringchr(s+1,c));
}
```

Esercizio 6
Si individuino opportuni dati di test per la seguente funzione,
evidenziando per quali dati si e' seguito l'approccio a scatola nera e per quali
quello a scatola trasparente.
Si dia una breve motivazione della scelta.

```
MinMass* MinMax (int* vett, int i, int j)
/* prec 0≤i<=j≤ numero degli elementi in vett  && vett != NULL;
postc: restituisce un puntatore a una struttura i cui campi interi contengono il
minimo e il massimo
* sugli elementi di vett tra vett[i] e vett[j], compresi*/
{MinMass *temp,*Mml,*Mmr;
int m;
temp =(MinMass*)malloc(sizeof(MinMass));
assert(temp);
assert(vett);
if(i == j) {temp -> min = vett[i];temp -> max = vett[i];return temp;}
m = (j+i)/2;
Mml  = MinMax(vett,i,m);
Mmr  = MinMax(vett,m+1,j);
temp -> max = max( Mml -> max, Mmr -> max);
temp -> min = min( Mml -> min, Mmr -> min);
return temp;}
```