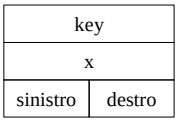
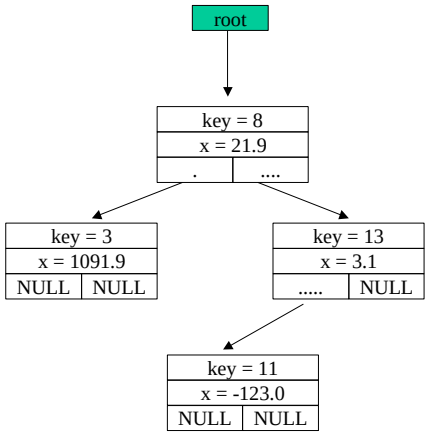
ESERCIZIO NUMERO 4 e SOLUZIONE

Data una struttura ad albero binario, implementata mediante la struttura dati NODO seguente:

```
typedef struct s_NODO{
        int             key;
        double          x;
        s_NODO          *destra;
        s_NODO          *sinistra;
}  NODO;
```





esempio di albero

Sia *root* la radice dell'albero (*NODO *root*), *root* e' una variabile locale del main, *root* assume valore NULL quando l'albero è vuoto, ogni nodo può avere due figli puntati dai puntatori destra e sinistra, che assumono valore NULL se il figlio corrispondente non esiste.
L'albero viene costruito allocando dinamicamente ciascun nodo chiamando la funzione *malloc*. L'albero può essere vuoto.
Quando l'albero non serve più, la memoria allocata per ciascun nodo deve essere rilasciata.
Per deallocare un'area di memoria si usa, come noto, la funzione di libreria standard *free(void*)* passandole come parametro l'indirizzo dell'area di memoria da deallocare.
Dopo la chiamata alla free() la memoria deallocata non puo' piu' essere acceduta.

Implementare la funzione void ***dealloca_albero***(NODO* *ppnodo);
che verrà chiamata ad. es. nel modo seguente:

```
main(){
        NODO *root;
        costruisci_albero( & root );
        usa_albero( root );
        dealloca_albero( & root );
}
```

La funzione ***dealloca_albero*** deve deallocare la memoria allocata per ciascuno dei nodi dell'albero, e infine deve porre a NULL il puntatore root di cui viene passato l'indirizzo come parametro della funzione, per far si che dopo la deallocazione l'albero appaia vuoto.

SOLUZIONE

```
void dealloca_albero(NODO* *ppnodo)
{
      if( *ppnodo )   {
            dealloca_albero( & ((*ppnodo)->sinistra)  );
            dealloca_albero( & ((*ppnodo)->destra)   );
            free(*ppnodo);
            *ppnodo=NULL;
      }
}
```