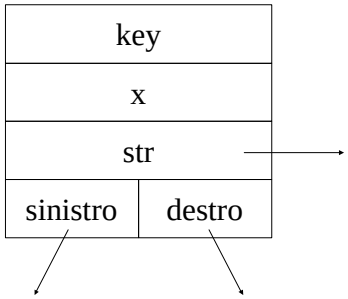
ESERCIZIO NUMERO 4

Sia data la struttura dati NODO seguente:

```
struct nodoalbero {
   int                  key;
   double               x;
   char                 *str;
   struct nodoalbero    *destra;
   struct nodoalbero    *sinistra;
   };
typedef struct nodoalbero NODO;
```



Tale struttura viene usata per implementare un albero binario (non bilanciato).
Il puntatore *str* serve a memorizzare una stringa (terminata dal \0) di lunghezza variabile, e quindi punta ad un'area di memoria, allocata dinamicamente al momento dell'inserimento del nodo dell'albero.
- Ogni nodo può avere due figli puntati dai puntatori *destra* e *sinistra*, che assumono valore NULL se il figlio corrispondente non esiste.
- I nodi sono ordinati secondo la chiave *key*. In particolare la chiave del figlio sinistro e' minore o uguale della chiave del padre, mentre la chiave del figlio destro e' maggiore della chiave del padre, e cosi' per tutti i nodi.
- Sia *root* la radice dell'albero *(NODO *root*)* una variabile LOCALE al main (vedi esempio).
- Si tenga presente che l'albero può essere vuoto (*root* puo' contenere un valore NULL) oppure contenere gia' alcuni nodi.

Si implementi una funzione

`void modifica_foglie(NODO *pnodo, double X, const char *STR);`

Tale funzione deve modificare il contenuto delle **foglie dell'albero** puntato dal parametro *pnodo*, scrivendo nel campo *x* di ogni foglia il valore *X*, e facendo il necessario per memorizzare, mediante il puntatore *str* di ogni foglia, il contenuto della stringa passata mediante il parametro *STR*.
Si noti che:

- la stringa puntata da *STR* potrebbe essere eliminata dopo la chiamata alla funzione ***`modifica_foglie`*** (vedi esempio) e quindi l'area di memoria puntata da *STR* deve essere utilizzata solo dentro la funzione ***`modifica_foglie`*** per copiare i caratteri in essa contenuti.
- la stringa *STR* ha lunghezza variabile e quindi potenzialmente diversa dalla lunghezza della stringa puntata dal campo *str* della foglia.

Si consideri anche che l'albero può essere vuoto (*root* puo' contenere un valore NULL) oppure contenere gia' alcuni nodi.

Qui di seguito è riportato un esempio di come può essere chiamata la funzione da implementare:

```
void main(void)
{
        NODO *root = NULL;
        costruisci_albero(&root);
        .............
        if ( 1 )
        {
                char str_nuova[] = "nuovastringa";
                modifica_foglie( root,  130.5, str_nuova );
        }

}
```

SOLUZIONE dell'ESERCIZIO 4 (C)

```
void modifica_foglie(NODO *pnodo, double X, const char *STR)
{
      if(pnodo)    {

            if( (pnodo->sinistra==NULL) && (pnodo->destra==NULL) )
            // se sono in una foglia
            {
                  // libero la memoria della vecchia stringa
                  free(pnodo->str);
                  // alloco memoria per la nuova stringa, in particolare
                  // la lunghezza della nuova stringa piu un carattere per il \0
                  pnodo->str = malloc ( 1+ strlen(STR) );
                  if( ! (pnodo->str) )
                        {   fprintf(stderr,"calloc failure\n"); exit(1); }
                  // copio i caratteri della nuova stringa nel nodo
                  strcpy( pnodo->str, STR);
                  // copio il nuovo valore X nel nodo
                  pnodo->x = X;
            }
            else // se non sono in una foglia
            {
                  modifica_foglie(pnodo->sinistra,X,STR);
                  modifica_foglie(pnodo->destra,X,STR);
            }
      }
}
```