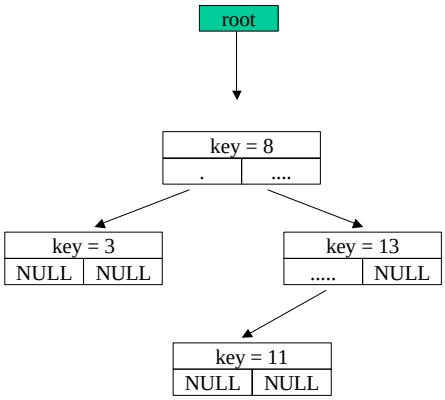
ESERCIZIO NUMERO 4

Sia data la struttura dati NODO seguente:

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

Tale struttura viene usata per implementare un albero binario (non bilanciato).
- Se root e' la radice dell'albero (NODO *root), 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.
- 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, come nell'esempio in figura.
- L'albero può essere vuoto.



Sia root la radice dell'albero (NODO *root) una variabile <u>LOCALE</u> al main (vedi esempio).
Si implementi una funzione void **inserisci(……)** che prevede alcuni parametri formali, tra cui un intero **int KEY** , ed **altri parametri di vostra scelta**.
Tale funzione deve **aggiungere all'albero puntato dalla variabile root un nuovo nodo**, avente come chiave key l'intero KEY, mantenendo l'ordine crescente dell'albero.
Per semplicita' si puo' ipotizzare che la allocazione con malloc o calloc abbia sempre esito positivo, quindi non e' richiesto di gestire un eventuale errore di allocazione.
Si consideri anche che l'albero può essere vuoto (root puo' contenere un valore NULL) oppure contenere gia' alcuni nodi.
Si ponga quindi particolare attenzione ai parametri passati alla funzione, per far si che dopo l'uscita dalla funzione l'albero risulti effettivamente modificato come richiesto.
Infine, nell'esempio riportato, fare un esempio di come e' chiamata la funzione per chiarire l'uso dei
es:

```
void main(void) {
        NODO *root = NULL;
         int       KEY ;
        …….
        KEY=19;
        inserisci ( KEY,    /* completare con gli  altri parametri necessari*/                    );
        KEY= -3;
        inserisci ( KEY ,   /* completare con gli  altri parametri necessari*/                    );
  }
```

SOLUZIONE dell'ESERCIZIO 4 (C)

```
void inserisci( int KEY , NODO* *ppnodo )
{
        if(  ! (*ppnodo)  )
        {
                *ppnodo = (NODO*) calloc ( 1 , sizeof(NODO) );
                (*ppnodo)->key=KEY;
                (*ppnodo)->destra   = NULL;
                (*ppnodo)->sinistra = NULL;
        }
        else
        {
                if ( KEY <= (*ppnodo)->key )
                        inserisci( KEY , & ((*ppnodo)->sinistra) ) ;
                else
                        inserisci( KEY , & ((*ppnodo)->destra) )  ;
                }
        }
}
```

esempio di chiamata:

```
  inserisci( KEY , &root ) ;
```