ESERCIZIO NUMERO 4 e SOLUZIONI

Data una struttura a lista, implementata mediante la struttura dati NODOLISTA seguente:

```
typedef struct nodolista {
        int         key;
        struct nodolista *next;
} NODOLISTA;
```

Sia *root* una variabile **locale** al main (dichiarata come: NODOLISTA *root), che funge da radice della lista, *root* è stata inizializzata al valore NULL, *root* assume valore NULL quando la lista è vuota.
Ogni elemento della lista e' di tipo NODOLISTA e può avere un successore puntato dal puntatore *next*, tale puntatore *next* assume valore NULL se il successore non esiste.
Gli elementi della lista sono ordinati in ordine crescente secondo il valore del campo *key*.
Implementare la funzione int ***elimina***(int KEY, NODOLISTA*** ***pradice);
che verrà chiamata ad. es. nel modo seguente:

```
main() {
        int ris;
        NODOLISTA *root=NULL;
        ....costruzione lista....
        .......
        ris = elimina ( 13,  &root );
        if ( ris==0) printf("elemento non trovato\n");
        .......
}
```

La funzione ***elimina*** deve togliere dalla lista, di cui viene passato il puntatore alla radice, l'elemento avente chiave KEY, deallocarlo e restituire 1. Se l'elemento con chiave KEY non esiste, la funzione restituisce 0.
All'uscita della funzione la variabile root deve ancora puntate al primo elemento della lista, se ne esiste ancora almeno uno, o assumere valore NULL. Ricordare che l'ultimo elemento della lista deve avere il campo *next* con valore NULL.

---

soluzione iterativa

```
int elimina(int KEY, NODOLISTA* *ppradice){
     while( (*ppradice!=NULL) && ((*ppradice)->key < KEY) )
           ppradice = &( (*ppradice)->next );
     if( (*ppradice!=NULL) && ((*ppradice)->key == KEY) ){
           NODOLISTA *next;
           next=(*ppradice)->next;
           free(*ppradice);
           *ppradice=next;
           return(1);
     }
     else  return(0);
}
```

---

soluzione ricorsiva

```
int elimina(int KEY, NODOLISTA* *ppradice){
     if( (*ppradice==NULL) || ((*ppradice)->key > KEY)) ) return(0);
       else {
           if( (*ppradice)->key == KEY ) {
                 NODOLISTA *next;
                 next=(*ppradice)->next;
                 free(*ppradice);
                 *ppradice=next;
                 return(1);
           }
       else  return( elimina(KEY, &((*ppradice)->next) )  );
       }
}
```