ESERCIZIO NUMERO 4 e SOLUZIONE

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 **globale** (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 ***aggiungi_in_ordine***(int KEY);
che verrà chiamata ad. es. nel modo seguente:

```
NODOLISTA *root=NULL;
main() {
int ris;   double X;
  .......
ris = aggiungi_in_ordine ( 13 );
if ( !ris) printf("errore di allocazione memoria\n");
ris = aggiungi_in_ordine ( 1 );
if ( !ris) printf("errore di allocazione memoria\n");
  .......
}
```

La funzione ***aggiungi_in_ordine*** deve inserire un nuovo elemento nella lista, **in posizione tale da mantenere l'ordine crescente**. La funzione deve creare il nuovo elemento di tipo NODOLISTA allocando dinamicamente spazio in memoria, deve inserire il valore KEY nel campo *key* di questo nuovo elemento, e deve inserire tale nuovo elemento nella lista puntata da *root.*
Ricordare che l'ultimo elemento della lista deve avere il campo *next* con valore NULL.
Al momento dell'allocazione dinamica dello spazio per il nuovo elemento, la funzione deve controllare se l'allocazione è stata possibile. Se la memoria non puo' essere allocata, la funzione deve restituire il valore 0. Se invece l'allocazione è stata possibile, alla fine del suo compito la funzione restituisce il valore 1.
All'uscita della funzione la variabile root deve ancora puntate al primo elemento della lista.

```
int aggiungi_in_ordine(int KEY){
      NODOLISTA* *ppnext;
      NODOLISTA  *ptr;

      ppnext=&root;
      while( (*ppnext!=NULL) && ((*ppnext)->key < KEY) )
            ppnext = &( (*ppnext)->next );
      ptr=(NODOLISTA*) calloc(1,sizeof(NODOLISTA));
      if(ptr) {
            ptr->key=KEY;
            ptr->next=*ppnext;
            *ppnext=ptr;
            return(1);
      }
      else
            return(0);
}
```