CdL in Scienze dell'Informazione, sede di Cesena

Sistemi per l'Elaborazione dell'Informazione 1

appello: 13 settembre 2000

ESERCIZIO NUMERO 4

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 inizializata 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.

Implementare la funzione int ***aggiungi_in_testa***(int KEY);
che verrà chiamata ad. es. nel modo seguente:

```
int ris, KEY;
double X;
KEY = 13;
  .......
ris = aggiungi_in_testa ( KEY );
if ( !ris) printf("errore di allocazione memoria\n");
```

La funzione ***aggiungi_in_testa*** deve allocare dinamicamente spazio in memoria per un nuovo elemento di tipo NODOLISTA, deve inserire il valore KEY nel campo key di questo nuovo elemento, e deve inserire tale nuovo elemento come primo elemento della lista puntata da root. Gli eventuali elementi gia' esistenti nella lista devono essere ovviamente accodati al nuovo elemento inserito.
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 immediatamente restituire il valore 0, e la lista deve rimanere nelle condizioni precedenti la chiamata della funzione. Se invece l'allocazione è stata possibile, alla fine del suo compito la funzione restituisce il valore 1.

SOLUZIONE dell'ESERCIZIO 4

```
int aggiungi_in_testa(int KEY){
        NODOLISTA *ptr;
        ptr=root;
        root= (NODOLISTA*) calloc(1,sizeof(NODOLISTA));
        if(root) {
                root->key=KEY;
                root->next=ptr;
                return(1);
        }
        else{
                root=ptr;
                return(0);
        }
}
```