Palindromi

Implementiamo un algoritmo per determinare se la parola contenuta in una lista concatenata è palindroma (parole che restano uguali se lette al contrario. Ad esempio `"anna"`, `"bob"` sono palindromi, `"ciao"` e `"bod"` non lo sono). Esistono vari modi per farlo, noi implementiamo un semplice algoritmo che crea una copia in ordine invertito della lista da controllare e verifica se la lista di origine e quella invertita son identiche.

Si chiede di implementare le seguenti funzioni:

- `void add(Nodo**lis, char c);`
- `int compareLists(Nodo*lis, Nodo* comp);`
- `void copyReversed(Nodo* lis, Nodo** copy);`
- `int checkPalindrome(Nodo*lis);`

Delle ultime due viene fornito lo pseudocodice:

Algorithm 1 copyReversed

Input: lis (una lista), copy (una lista vuota)
Output: Nessuno. **Side effect**: copy contiene una lista di lunghezza uguale a lis, in cui i nodi contengono gli stessi caratteri di quelli di lis ma in ordine inverso.
if *list* = nil **then**
  return
end if
copyReversed(lista a partire dal nodo successivo di lis, *copy*)
add(*copy*, carattere contenuto nel primo nodo di lis)

Algorithm 2 checkPalindrome

Input: lis (una lista)
Output: TRUE se la lista passata contiene una sequenza di caratteri palindroma, FALSE altrimenti
$inv \leftarrow$ nuova lista vuota
$copyReversed(lis, inv)$
return $compareLists(lis, inv)$

`add` serve ad ad aggiungere un nodo in coda alla lista `lis` contenente il carattere `c` come valore.

`compareLists` deve restituire True se `lis` e `comp` sono due liste identiche (hanno lo stesso numero di nodi e il contenuto dei nodi è identico). Esistono più modi di eseguire questo controllo ma si consiglia di usare la ricorsione.

Se necessario, si possono aggiungere funzioni di supporto, ma è necessario che le funzioni richieste siano implementate esattamente con i prototipi forniti.

> Lo pseudocodice non è codice completo in C: non considera i tipi dei vari oggetti, l'uso di puntatori/riferimenti, e dunque dell'operatore "->" anziché l'operatore ".": 'e lasciato allo studente il compito di determinare questi dettagli come adeguato.

Soluzione (fare click per visualizzare)

La soluzione proposta è la seguente:

```c
void add(Nodo **lista, char c)
{
    if (lista == NULL)
    {
        printf("Passare per riferimento ad ADD\n");
    }
    if (*lista == NULL)
    {
        *lista = malloc(sizeof(Nodo));
        assert(*lista != NULL);
        (*lista)->value = c;
        (*lista)->next = NULL;
    }
    else
    {
        add(&(*lista)->next, c);
    }
}

void copyReversed(Nodo *src, Nodo **copy)
{
    if (copy == NULL)
    {
        printf("Passare la lista di destinazione di copia per puntatore");
    }
    if (src == NULL)
    {
        return;
    }
    copyReversed(src->next, copy);
    add(copy, src->value);
}

int compareLists(Nodo *list_a, Nodo *list_b)
{
    if (list_a == NULL)
    {
        if (list_b == NULL)
        {
            return 1;
        }
        else
        {
            return 0;
        }
    }
    if (list_a->value != list_b->value)
    {
        return 0;
    }
    else
    {
        return compareLists(list_a->next, list_b->next);
    }
}

int checkPalindrome(Nodo *lista)
{
    if (lista == NULL)
```