ISTRUZIONI GENERALI

Spedire un file .c contenente esclusivamente le implementazioni delle funzioni descritte di seguito (ed eventuali metodi ausiliari necessari per la corretta compilazione ed esecuzione). Non includere il main e non inviare alcun file .h. Per l'esercizio 3, usare i tipi per le liste introdotti a lezione (saranno quelli che useremo per la correzione automatica). Tutte le soluzioni devono essere ricorsive.

ESERCIZIO 1
Data una stringa s, calcolare una nuova stringa in cui tutte le lettere 'a' minuscole con un accento acuto o grave sono rimpiazzate dalla 'a' senza accento

```
char *rimpiazzaAccenti(char *s);
```

Ad esempio: rimpiazzaAccenti ("papà") = "papa"
rimpiazzaAccenti ("papå") = "papå"

ESERCIZIO 2
Verificare se una stringa s1 è sottostringa di s2: int sottostringa (char *s1, char *s2);
Una sottostringa è un insieme di caratteri in posizioni consecutive.

Ad esempio: sottostringa ("pippo", "apippop") = 1
sottostringa ("pippo", "apipppo") = 0
sottostringa ("pippo", "apoppop") = 0
sottostringa ("", "pippo") = 1

ESERCIZIO 3
Dati una lista L ed un intero k>0, rimuovere da L un elemento ogni k (ovvero, rimuovere gli elementi in posizione k, 2k, 3k, etc, assumendo che le posizioni siano numerate a partire da 1). La lista L va modificata. E' ammesso fare deallocazioni, ma non allocazioni di nuova memoria.

```
void eliminaOgniK(ListPtr *L, int k);
```

Ad esempio: eliminaOgniK ("irenefinocchi", 1) = ""
eliminaOgniK ("irenefinocchi", 2) = "ieeioci"
eliminaOgniK ("irenefinocchi", 3) = "irneincci"

Il metodo deve usare una sottoprocedura ricorsiva con il seguente prototipo:

```
void eliminaOgniKRec(ListPtr *L, int h, int k);
```

La sottoprocedura elimina un elemento ogni k, esclusi i primi h elementi della lista.