Esercizi stringhe e funzioni stringhe

Si propongono alcuni esercizi riguardo l'impiego di funzionalità legate alle stringhe C.

1. Scrivere una funzione che individui la posizione di una stringa all'interno di due stringhe, e verifichi se queste due stringhe sono uguali dalla posizione individuata in poi. La funzione rispetti il seguente prototipo:

   ```
   int suffix_compare(const char *s1, const char *s2, const char *st);
   ```

   dove:

   - `s1` e `s2` sono i puntatori alle due stringhe di input;
   - `st` è la sotto-stringa da individuare all'interno delle due stringhe precedenti;
   - la funzione ritorna `1` se il confronto ha successo, oppure `0` se le due stringhe sono differenti o se la stringa puntata da `st` non è presente in alcuna posizione di `s1` o `s2`.

2. Realizzare una funzione che, data una stringa e un carattere, individui il carattere indicato e restituisca la sotto-stringa della stringa originale a partire dalla posizione del carattere. In caso di molteplici caratteri uguali, si scelga il carattere con indice più alto (ovvero, più a "destra" nella stringa). Ad esempio:

   - data la stringa `"Hello, World!"` e il carattere `'W'`, la funzione restituisce "World!";
   - data la stringa `"Hello, World!"` e il carattere `'l'`, la funzione restituisce "ld!".

   La funzione deve rispettare il seguente prototipo:

   ```
   char* mysubstring(const char *s, char symbol);
   ```

   dove:

   - `s` è il puntatore alla stringa da ispezionare;
   - `symbol` è il carattere da cercare. Assumere che il carattere ricevuto sia sempre valido;
   - la funzione ritorna il puntatore a una nuova area di memoria che contiene la sotto-stringa individuata. Nel caso in cui non esista il carattere richiesto nella stringa (compreso il caso di avere in input una stringa vuota)o la funzione non riesca ad allocare la memoria necessaria la funzione ritorna NULL.

3. Realizzare una funzione per la conversione di una stringa che rappresenta una lista di numeri interi separati davirgola in un array di interi. La funzione deve rispettare il seguente prototipo:

   ```
   int stringsplit(long **values, const char *s);
   ```

   La funzione si aspetta che la variabile `s` punti a una stringa C che rappresenta una lista di numeri interi (sia positivi, sia negativi) separati da virgola (ovvero, il carattere ','). La funzione deve convertire la stringa in unarray di variabili long, che restituisce tramite la variabile values. La funzione deve gestire l'allocazione dellamemoria necessaria per memorizzare l'array in maniera opportuna. Ad esempio, per una stringa con valore `"5,7,-10"`, la funzione restituirà in modo opportuno l'array `{5,7,-10}`. La funzione deve gestire i casi di esecuzione anomala nelle seguenti modalità:

   - se la stringa è vuota, la funzione imposta in modo opportuno il valore di values al puntatore speciale NULLe restituisce come valore di ritorno 0;
   - se la funzione riscontra degli errori di conversione restituisce valore -1. Ad esempio, stringhe con valori non numerici, valori mancanti, ecc.;
   - se la funzione riscontra degli errori di esecuzione (e.g., allocazione memoria) restituisce valore -2.

4. Scrivere una funzione che, data una stringa C contenente una matrice di valori interi in formato tabulare, in cui `'\t'` è il carattere che delimita le colonne e il carattere `'\n'` delimita le righe, crea una matrice di variabili `int`. Le dimensioni della matrice sono note. Rispettare il seguente prototipo:

```
int convert_matrix(unsigned rows_num, unsigned cols_num, int r[][cols_num], const char *s);
```

dove:

- `rows_num` e `cols_num` identificano le dimensioni della matrice;
- `r` è il puntatore all'area di memoria della matrice, già allocata, in cui vengono salvati i valori;
- `s` è il puntatore alla stringa C di input;
- la funzione restituisce valore `1` se la conversione ha avuto successo, ovvero se la matrice presente nella stringa ha effettivamente le dimensioni richieste e il formato corretto, altrimenti `0`.

5. Scrivere una funzione simile a quella dell'esercizio precedente che, data una stringa C contenente dei valori interi in formato tabulare, li converte in valori interi di variabili `int`. In questo caso le dimensioni della struttura non sono note a priori, e il numero di valori presenti in ogni riga può essere differente. Rispettare il seguente prototipo:

```
typedef struct int_list {
    unsigned size;
    unsigned *sizes;
    int **values;
} int_list_t;

long convert_var_matrix(int_list_t *r, const char*s);
```

dove:

- `r` è il puntatore ai valori individuati, che punta ad un'area di memoria già allocata con la struct di tipo `int_list_t`. I puntatori interni alla struct invece non puntano ad aree di memoria già allocate. Nel caso in cui la stringa non include valori, la struttura viene comunque modificata impostando a `0` la variabile `size` e a `NULL` i puntatori;
- `s` è il puntatore alla stringa C di input;
- la funzione restituisce il numero totale di valori individuati. Nel caso in cui la stringa non include valori, il valore di ritorno è `0`.