Esercizi malloc

Considerare alcuni degli esercizi proposti nella parte 3 (array e stringe). Si propone di creare delle funzioni simili, ma che hanno un prototipo differente per gestire dinamicamente l'allocazione della memoria al loro interno.

1. Implementare una funzione che inverta l'ordine dei valori di un array di dati interi che rispetti il seguente prototipo:

   ```
   int* reversei(const int *values, unsigned size);
   ```

   dove:

   - `values` è il puntatore all'array di input
   - `size` è la dimensione dell'array
   - la funzione ritorna il puntatore all'array invertito, allocato dalla funzione (`NULL` in caso di errore di allocazione)

2. Implementare una funzione che inverta l'ordine dei caratteri di una stringa C che rispetti il seguente prototipo:

   ```
   void reverses(char **r, const char *s);
   ```

   dove:

   - `r` gestisce il puntatore per memorizzare la stringa C invertita, fa in modo di restituire NULL in caso di errore di allocazione;
   - `s` è il puntatore alla stringa in input

3. Implementare una funzione che dati due array di valori interi ordinati, generi un terzo array che contenga tutti i valori dei precedenti array in modo ordinato. Assumere che all'interno degli array ci possano essere elementi duplicati. La funzione rispetti il seguente prototipo:

   ```
   void merge(int **r, const int *a1, unsigned s1, const int *a2, unsigned s2);
   ```

   dove:

   - `r` è il puntatore all'array generato, della dimensione opportuna per contenere esattamente le due stringhe. Ha valore `NULL` in caso di errore di allocazione
   - `a1` è il puntatore al primo array di input
   - `s1` è la dimensione del primo array
   - `a2` è il puntatore al secondo array di input
   - `s2` è la dimensione del secondo array