Esame 20220111

Esercizio 2

(1) **Esercizio 2 v1**

ESSAY | marked out of 10 | penalty 0 | File picker

Scrivere la dichiarazione e la definizione di una procedura **ricorsiva** `compute_lists`, che prende come argomento una stringa (letta da tastiera) di lunghezza non nota e che non puó essere modificata (i.e. `const`), ma ben formata (i.e. con carattere di terminatore `'\0'`), e due liste concatenate `s1` e `s2` di tipo `node *` passate entrambe per riferimento. La procedura ricorsiva deve fare le seguenti operazioni:

- Ogni carattere compreso tra `'A'` e `'M'` (inclusi) viene trasformato in modo che `'A'` diventi `'m'`, `'B'` diventi `'l'`, ..., `'L'` diventi `'b'`, e `'M'` diventi `'a'`, il carattere risultante viene inserito nella lista `s1` in modo che l'ordine in cui si incontrano rispetti l'ordine della stringa di partenza;
- Ogni carattere compreso tra `'N'` e `'Z'` (inclusi) viene trasformato in modo che `'N'` diventi `'z'`, `'O'` diventi `'y'`, ..., `'Y'` diventi `'o'`, e `'Z'` diventi `'n'`, il carattere risultante viene inserito nella lista `s2` in modo che l'ordine in cui si incontrano rispetti l'ordine della stringa di partenza;
- Ogni carattere che non soddisfa le regole di cui sopra é ignorato.

La procedura `compute_list` può solo chiamare se stessa, e solo con lo stesso numero di argomenti (NON è consentito fare overloading e/o utilizzare argomenti addizionali inizializzati nella dichiarazione/definizione). Sono solo consentite (se ritenute necessarie) chiamate a funzioni semplici per facilitare/rendere modulare la conversione di un carattere in altro carattere per realizzare le condizioni di cui sopra. La conversione dei caratteri deve essere realizzata SOLO mediante operazioni aritmetiche sui caratteri. NON sono ammesse soluzioni che usano case/switch/if per effettuare le conversioni di cui sopra.

La funzione è inserita in un semplice programma che legge una stringa da tastiera, costruisce le due liste concatenate, le stampa, e dealloca le liste create. Il `main`, le funzioni `stampa_lista`, e `dealloca` NON devono essere modificate. Alcuni esempi di esecuzione sono i seguenti:

```
computer > ./a.out "-=abcdefghijklmnopqrstuvwxyz0123456789."
La stringa da analizzare e': -=abcdefghijklmnopqrstuvwxyz0123456789.
Lista s1:
Lista s2:
computer > ./a.out "-=ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789."
La stringa da analizzare e': -=ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789.
Lista s1: m l k j i h g f e d c b a
Lista s2: z y x w v u t s r q p o n
```

Note:

- Scaricare il file `esercizio2.cc`, modificarlo per inserire la dichiarazione e la definizione della funzione `compute_list`, e caricare il file sorgente risultato delle vostre modifiche a soluzione di questo esercizio nello spazio apposito.

- La funzione `compute_list` deve essere ricorsiva ed al suo interno **NON ci possono essere cicli o chiamate a funzioni contenenti cicli**. Si può però fare uso di funzioni ausiliarie da chiamare all'interno di questa funzione che NON contengano cicli secondo le restrizioni specificate sopra.
- Le uniche assunzioni che si possono fare sull'input e su dimensioni di eventuali strutture/array utilizzate nel file di partenza fornito sono **solo quelle espressamente specificate in questo testo** (e NON quelle riportate nel file fornito, che sono SOLO indicative per consentire di svolgere l'esame).
- All'interno di questo programma **non è ammesso** l'utilizzo di variabili globali o di tipo `static` e di funzioni di libreria al di fuori di quelle definite in `cstddef` o `cstdlib`.
- Si ricorda che, l'esempio di esecuzione è puramente indicativo, e la soluzione proposta NON deve funzionare solo per l'input fornito, ma deve essere robusta a variazioni compatibili con la specifica riportata in questo testo.

Information for graders:

(2) **Esercizio 2 v2**

ESSAY | marked out of 10 | penalty 0 | File picker

Scrivere la dichiarazione e la definizione di una procedura **ricorsiva** `compute_lists`, che prende come argomento una stringa (letta da tastiera) di lunghezza non nota e che non puó essere modificata (i.e. `const`), ma ben formata (i.e. con carattere di terminatore `'\0'`), e due liste concatenate `s1` e `s2` di tipo `node *` passate entrambe per riferimento. La procedura ricorsiva deve fare le seguenti operazioni:

- Ogni carattere compreso tra `'a'` e `'m'` (inclusi) viene trasformato in modo che `'a'` diventi `'M'`, `'b'` diventi `'L'`, ..., `'l'` diventi `'B'`, e `'m'` diventi `'A'`, il carattere risultante viene inserito nella lista `s1` in modo che l'ordine in cui si incontrano rispetti l'ordine della stringa di partenza;
- Ogni carattere compreso tra `'n'` e `'z'` (inclusi) viene trasformato in modo che `'n'` diventi `'Z'`, `'o'` diventi `'Y'`, ..., `'y'` diventi `'O'`, e `'z'` diventi `'N'`, il carattere risultante viene inserito nella lista `s2` in modo che l'ordine in cui si incontrano rispetti l'ordine della stringa di partenza;
- Ogni carattere che non soddisfa le regole di cui sopra é ignorato.

La procedura `compute_list` può solo chiamare se stessa, e solo con lo stesso numero di argomenti (NON è consentito fare overloading e/o utilizzare argomenti addizionali inizializzati nella dichiarazione/definizione). Sono solo consentite (se ritenute necessarie) chiamate a funzioni semplici per facilitare/rendere modulare la conversione di un carattere in altro carattere per realizzare le condizioni di cui sopra. La conversione dei caratteri deve essere realizzata SOLO mediante operazioni aritmetiche sui caratteri. NON sono ammesse soluzioni che usano case/switch/if per effettuare le conversioni di cui sopra.

La funzione è inserita in un semplice programma che legge una stringa da tastiera, costruisce le due liste concatenate, le stampa, e dealloca le liste create. Il `main`, le funzioni `stampa_lista`, e `dealloca` NON devono essere modificate. Alcuni esempi di esecuzione sono i seguenti:

```
computer > ./a.out "-=ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789."
La stringa da analizzare e': -=ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789.
Lista s1:
Lista s2:
computer > ./a.out "-=abcdefghijklmnopqrstuvwxyz0123456789."
La stringa da analizzare e': -=abcdefghijklmnopqrstuvwxyz0123456789.
Lista s1: M L K J I H G F E D C B A
Lista s2: Z Y X W V U T S R Q P O N
```

Note:

- Scaricare il file `esercizio2.cc`, modificarlo per inserire la dichiarazione e la definizione della funzione `compute_list`, e caricare il file sorgente risultato delle vostre modifiche a soluzione di questo esercizio nello spazio apposito.
- La funzione `compute_list` deve essere ricorsiva ed al suo interno **NON ci possono essere cicli o chiamate a funzioni contenenti cicli**. Si può però fare uso di funzioni ausiliarie da chiamare all'interno di questa funzione che NON contengano cicli secondo le restrizioni specificate sopra.

- Le uniche assunzioni che si possono fare sull'input e su dimensioni di eventuali strutture/array utilizzate nel file di partenza fornito sono **solo quelle espressamente specificate in questo testo** (e NON quelle riportate nel file fornito, che sono SOLO indicative per consentire di svolgere l'esame).
- All'interno di questo programma **non è ammesso** l'utilizzo di variabili globali o di tipo `static` e di funzioni di libreria al di fuori di quelle definite in `cstddef` o `cstdlib`.
- Si ricorda che, l'esempio di esecuzione è puramente indicativo, e la soluzione proposta NON deve funzionare solo per l'input fornito, ma deve essere robusta a variazioni compatibili con la specifica riportata in questo testo.

Information for graders:

(3) **Esercizio 2 v3**

ESSAY | marked out of 10 | penalty 0 | File picker

Scrivere la dichiarazione e la definizione di una procedura **ricorsiva** `compute_lists`, che prende come argomento una stringa (letta da tastiera) di lunghezza non nota e che non puó essere modificata (i.e. `const`), ma ben formata (i.e. con carattere di terminatore `'\0'`), e due liste concatenate `s1` e `s2` di tipo `node *` passate entrambe per riferimento. La procedura ricorsiva deve fare le seguenti operazioni:

- Ogni carattere compreso tra `'0'` e `'4'` (inclusi) viene trasformato in modo che `'0'` diventi `'4'`, `'1'` diventi `'3'`, `'2'` diventi `'2'`, il carattere risultante viene inserito nella lista `s1` in modo che l'ordine in cui si incontrano rispetti l'ordine della stringa di partenza;
- Ogni carattere compreso tra `'5'` e `'9'` (inclusi) viene trasformato in modo che `'5'` diventi `'9'`, `'6'` diventi `'8'`, `'7'` diventi `'7'`, il carattere risultante viene inserito nella lista `s2` in modo che l'ordine in cui si incontrano rispetti l'ordine della stringa di partenza;
- Ogni carattere che non soddisfa le regole di cui sopra é ignorato.

La procedura `compute_list` può solo chiamare se stessa, e solo con lo stesso numero di argomenti (NON è consentito fare overloading e/o utilizzare argomenti addizionali inizializzati nella dichiarazione/definizione). Sono solo consentite (se ritenute necessarie) chiamate a funzioni semplici per facilitare/rendere modulare la conversione di un carattere in altro carattere per realizzare le condizioni di cui sopra. La conversione dei caratteri deve essere realizzata SOLO mediante operazioni aritmetiche sui caratteri. NON sono ammesse soluzioni che usano case/switch/if per effettuare le conversioni di cui sopra.

La funzione è inserita in un semplice programma che legge una stringa da tastiera, costruisce le due liste concatenate, le stampa, e dealloca le liste create. Il `main`, le funzioni `stampa_lista`, e `dealloca` NON devono essere modificate. Alcuni esempi di esecuzione sono i seguenti:

```
computer > ./a.out "-=abcdefghijKLMNOPQRSTUVWXYZ."
La stringa da analizzare e': -=abcdefghijKLMNOPQRSTUVWXYZ.
Lista s1:
Lista s2:
computer> ./a.out "-=abcdefghijklmnopqrstuvwxyz0123456789."
La stringa da analizzare e': -=abcdefghijklmnopqrstuvwxyz0123456789.
Lista s1: 4 3 2 1 0
Lista s2: 9 8 7 6 5
```

Note:

- Scaricare il file `esercizio2.cc`, modificarlo per inserire la dichiarazione e la definizione della funzione `compute_list`, e caricare il file sorgente risultato delle vostre modifiche a soluzione di questo esercizio nello spazio apposito.
- La funzione `compute_list` deve essere ricorsiva ed al suo interno **NON ci possono essere cicli o chiamate a funzioni contenenti cicli**. Si può però fare uso di funzioni ausiliarie da chiamare all'interno di questa funzione che NON contengano cicli secondo le restrizioni specificate sopra.
- Le uniche assunzioni che si possono fare sull'input e su dimensioni di eventuali strutture/array utilizzate nel file di partenza fornito sono **solo quelle espressamente specificate in questo testo** (e NON quelle riportate nel file fornito, che sono SOLO indicative per consentire di svolgere l'esame).

- All'interno di questo programma **non è ammesso** l'utilizzo di variabili globali o di tipo `static` e di funzioni di libreria al di fuori di quelle definite in `cstddef` o `cstdlib`.
- Si ricorda che, l'esempio di esecuzione è puramente indicativo, e la soluzione proposta NON deve funzionare solo per l'input fornito, ma deve essere robusta a variazioni compatibili con la specifica riportata in questo testo.

Information for graders:

(4) **Esercizio 2 v4**

ESSAY | marked out of 10 | penalty 0 | File picker

Scrivere la dichiarazione e la definizione di una procedura **ricorsiva** `compute_lists`, che prende come argomento una stringa (letta da tastiera) di lunghezza non nota e che non puó essere modificata (i.e. `const`), ma ben formata (i.e. con carattere di terminatore `'\0'`), e due liste concatenate `s1` e `s2` di tipo `node *` passate entrambe per riferimento. La procedura ricorsiva deve fare le seguenti operazioni:

- Ogni carattere compreso tra `'C'` e `'N'` (inclusi) viene trasformato in modo che `'C'` diventi `'n'`, `'D'` diventi `'m'`, ..., `'M'` diventi `'d'`, e `'N'` diventi `'c'`, il carattere risultante viene inserito nella lista `s1` in modo che l'ordine in cui si incontrano rispetti l'ordine della stringa di partenza;
- Ogni carattere compreso tra `'O'` e `'Z'` (inclusi) viene trasformato in modo che `'O'` diventi `'z'`, `'P'` diventi `'y'`, ..., `'Y'` diventi `'p'`, e `'Z'` diventi `'o'`, il carattere risultante viene inserito nella lista `s2` in modo che l'ordine in cui si incontrano rispetti l'ordine della stringa di partenza;
- Ogni carattere che non soddisfa le regole di cui sopra é ignorato.

La procedura `compute_list` può solo chiamare se stessa, e solo con lo stesso numero di argomenti (NON è consentito fare overloading e/o utilizzare argomenti addizionali inizializzati nella dichiarazione/definizione). Sono solo consentite (se ritenute necessarie) chiamate a funzioni semplici per facilitare/rendere modulare la conversione di un carattere in altro carattere per realizzare le condizioni di cui sopra. La conversione dei caratteri deve essere realizzata SOLO mediante operazioni aritmetiche sui caratteri. NON sono ammesse soluzioni che usano case/switch/if per effettuare le conversioni di cui sopra.

La funzione è inserita in un semplice programma che legge una stringa da tastiera, costruisce le due liste concatenate, le stampa, e dealloca le liste create. Il `main`, le funzioni `stampa_lista`, e `dealloca` NON devono essere modificate. Alcuni esempi di esecuzione sono i seguenti:

```
computer > ./a.out "-=abcdefghijklmnopqrstuvwxyz0123456789."
La stringa da analizzare e': -=abcdefghijklmnopqrstuvwxyz0123456789.
Lista s1:
Lista s2:
computer > ./a.out "-=ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789."
La stringa da analizzare e': -=ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789.
Lista s1: n m l k j i h g f e d c
Lista s2: z y x w v u t s r q p o
```

Note:

- Scaricare il file `esercizio2.cc`, modificarlo per inserire la dichiarazione e la definizione della funzione `compute_list`, e caricare il file sorgente risultato delle vostre modifiche a soluzione di questo esercizio nello spazio apposito.
- La funzione `compute_list` deve essere ricorsiva ed al suo interno **NON ci possono essere cicli o chiamate a funzioni contenenti cicli**. Si può però fare uso di funzioni ausiliarie da chiamare all'interno di questa funzione che NON contengano cicli secondo le restrizioni specificate sopra.

- Le uniche assunzioni che si possono fare sull'input e su dimensioni di eventuali strutture/array utilizzate nel file di partenza fornito sono **solo quelle espressamente specificate in questo testo** (e NON quelle riportate nel file fornito, che sono SOLO indicative per consentire di svolgere l'esame).
- All'interno di questo programma **non è ammesso** l'utilizzo di variabili globali o di tipo `static` e di funzioni di libreria al di fuori di quelle definite in `cstddef` o `cstdlib`.
- Si ricorda che, l'esempio di esecuzione è puramente indicativo, e la soluzione proposta NON deve funzionare solo per l'input fornito, ma deve essere robusta a variazioni compatibili con la specifica riportata in questo testo.

Information for graders:

Total of marks: 40