Esame 20220223

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_list`, che prende come argomento una lista concatenata `l` di tipo `List *` in cui il campo `info` é un intero, che utilizza `NULL` come terminatore della lista e due liste concatenate `s1` e `s2` di tipo `List *` **passate entrambe per riferimento** (e che non sono state inizializzate a nessun valore). La procedura ricorsiva scorre la lista `l` e deve fare le seguenti operazioni:

- Se il campo `info` del nodo corrente è pari e non è un multiplo di 3 lo memorizza in `s1` rispettando l'ordine della lista di partenza;
- Se il campo `info` del nodo corrente è un multiplo di 3 lo memorizza in `s2` rispettando l'ordine della lista di partenza;;
- Altrimenti il nodo corrente deve essere ignorato.
- L'ultimo elemento delle liste `s1` e `s2` è un nodo che continente il numero di elementi inseriti nella lista rispettiva.

La procedura `compute_list` **deve essere ricorsiva**, Sono consentite (se ritenute necessarie) chiamate a funzioni ricorsive che **non** contengano iterazioni esplicite (for, while, do).

La funzione è inserita in un semplice programma che genera una lista concatenata in modo random chiama la funzione da definire, stampa le liste costruite e le de-alloca, e che non deve essere modificato. L'esecuzione (se non si abilita l'esecuzione random) é la seguente:

```
computer > ./a.out
Lista iniziale : "33" "36" "27" "15" "43" "35" "36" "42" "49" "21" "12" "27"
Lista l1 : "40" "26" "40" "26" "22" "32" "2" "22" "8" "34" "20" "11"
Lista l2 : "33" "36" "27" "15" "36" "42" "21" "12" "27" "9" "36" "18" "30" "
```

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.
- 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`, `cstdlib`, `iostream`.
- 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_list`, che prende come argomento una lista concatenata `l` di tipo `List *` in cui il campo `info` é un intero, che utilizza `NULL` come terminatore della lista e due liste concatenate `s1` e `s2` di tipo `List *` **passate entrambe per riferimento** (e che non sono state inizializzate a nessun valore). La procedura ricorsiva scorre la lista `l` e deve fare le seguenti operazioni:

- Se il campo `info` del nodo corrente è pari e non è un multiplo di 3 lo memorizza in `s2` rispettando l'ordine della lista di partenza;
- Se il campo `info` del nodo corrente è un multiplo di 3 lo memorizza in `s1` rispettando l'ordine della lista di partenza;;
- Altrimenti il nodo corrente deve essere ignorato.
- L'ultimo elemento delle liste `s1` e `s2` è un nodo che continente il numero di elementi inseriti nella lista rispettiva.

La procedura `compute_list` **deve essere ricorsiva**, Sono consentite (se ritenute necessarie) chiamate a funzioni ricorsive che **non** contengano iterazioni esplicite (for, while, do).

La funzione è inserita in un semplice programma che genera una lista concatenata in modo random chiama la funzione da definire, stampa le liste costruite e le de-alloca, e che non deve essere modificato. L'esecuzione (se non si abilita l'esecuzione random) é la seguente:

```
computer > ./a.out
Lista iniziale : "33" "36" "27" "15" "43" "35" "36" "42" "49" "21" "12" "27"
Lista l1 : "33" "36" "27" "15" "36" "42" "21" "12" "27" "9" "36" "18" "30" "
Lista l2 : "40" "26" "40" "26" "22" "32" "2" "22" "8" "34" "20" "11"
```

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.
- 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`, `cstdlib`, `iostream`.
- 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_list`, che prende come argomento una lista concatenata `l` di tipo `List *` in cui il campo `info` é un intero, che utilizza `NULL` come terminatore della lista e due liste concatenate `s1` e `s2` di tipo `List *` **passate entrambe per riferimento** (e che non sono state inizializzate a nessun valore). La procedura ricorsiva scorre la lista `l` e deve fare le seguenti operazioni:

- Se il campo `info` del nodo corrente è un multiplo di 3 e non è pari lo memorizza in `s1` rispettando l'ordine della lista di partenza;
- Se il campo `info` del nodo corrente è un numero pari lo memorizza in `s2` rispettando l'ordine della lista di partenza;;
- Altrimenti il nodo corrente deve essere ignorato.
- L'ultimo elemento delle liste `s1` e `s2` è un nodo che continente il numero di elementi inseriti nella lista rispettiva.

La procedura `compute_list` **deve essere ricorsiva**, Sono consentite (se ritenute necessarie) chiamate a funzioni ricorsive che **non** contengano iterazioni esplicite (for, while, do).

La funzione è inserita in un semplice programma che genera una lista concatenata in modo random chiama la funzione da definire, stampa le liste costruite e le de-alloca, e che non deve essere modificato. L'esecuzione (se non si abilita l'esecuzione random) é la seguente:

```
computer > ./a.out
Lista iniziale : "33" "36" "27" "15" "43" "35" "36" "42" "49" "21" "12" "27"
Lista l1 : "33" "27" "15" "21" "27" "9" "21" "15" "8"
Lista l2 : "36" "36" "42" "12" "40" "26" "40" "26" "22" "36" "18" "32" "30"
```

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.
- 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`, `cstdlib`, `iostream`.
- 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_list`, che prende come argomento una lista concatenata `l` di tipo `List *` in cui il campo `info` é un intero, che utilizza `NULL` come terminatore della lista e due liste concatenate `s1` e `s2` di tipo `List *` **passate entrambe per riferimento** (e che non sono state inizializzate a nessun valore). La procedura ricorsiva scorre la lista `l` e deve fare le seguenti operazioni:

- Se il campo `info` del nodo corrente è un multiplo di 3 e non è pari lo memorizza in `s2` rispettando l'ordine della lista di partenza;
- Se il campo `info` del nodo corrente è un numero pari lo memorizza in `s1` rispettando l'ordine della lista di partenza;;
- Altrimenti il nodo corrente deve essere ignorato.
- L'ultimo elemento delle liste `s1` e `s2` è un nodo che continente il numero di elementi inseriti nella lista rispettiva.

La procedura `compute_list` **deve essere ricorsiva**, Sono consentite (se ritenute necessarie) chiamate a funzioni ricorsive che **non** contengano iterazioni esplicite (for, while, do).

La funzione è inserita in un semplice programma che genera una lista concatenata in modo random chiama la funzione da definire, stampa le liste costruite e le de-alloca, e che non deve essere modificato. L'esecuzione (se non si abilita l'esecuzione random) é la seguente:

```
computer > ./a.out
Lista iniziale : "33" "36" "27" "15" "43" "35" "36" "42" "49" "21" "12" "27"
Lista l1 : "36" "36" "42" "12" "40" "26" "40" "26" "22" "36" "18" "32" "30"
Lista l2 : "33" "27" "15" "21" "27" "9" "21" "15" "8"
```

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.
- 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`, `cstdlib`, `iostream`.
- 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