Esame 20220617

Esercizio 2

(1) **Esercizio 2 v1**

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

Scrivere la dichiarazione e la definizione di una funzione **ricorsiva** `compute_sum`, che prende come argomento un intero *positivo* `num` e restituisce:

- la somma delle cifre del numero se questa somma è minore di 10;
- il risultato della riapplicazione della funzione sulla somma calcolata addizionata di 10 se la somma calcolata è pari;
- il risultato della riapplicazione della funzione sulla somma calcolata addizionata di 1 negli altri casi.

Per evitare ambiguità, riportiamo qui di seguito un esempio di applicazione delle regole qui sopra specificate. Sia `sum` una funzione che calcola la somma delle cifre che compongono il numero, `sum(99019) = 28` che è maggiore di 10 e pari, quindi sommo 10, `sum(38) = 11` che é maggiore di 10 e dispari, quindi sommo 1, `sum(12) = 3`, minore di 10 e quindi `compute_sum(99019) = 3`.

La funzione `compute_sum` **deve essere ricorsiva** e **NON deve contenere iteratori** espliciti (`for`, `while`, `do-while`). Sono **solo** consentite (se ritenute necessarie) chiamate a funzioni ricorsive ausiliarie che a loro volta **non** contengano iterazioni esplicite (`for`, `while`, `do-while`).

La funzione è inserita in un semplice programma che legge un intero positivo come unico argomento a linea di comando, lo converte in intero, chiama la funzione `compute_sum`, stampa a video il numero specificato ed il risultato della chiamata a `compute_sum`. Alcuni esempi di esecuzione sono i seguenti:

```
computer > ./a.out 38
The initial integer is: 38
The value of compute_sum(38) = 3
computer > ./a.out 99019
The initial integer is: 99019
The value of compute_sum(99019) = 3
computer > ./a.out 91019
The initial integer is: 91019
The value of compute_sum(91019) = 3
computer > ./a.out 191019
The initial integer is: 191019
The value of compute_sum(191019) = 4
computer > ./a.out 491019
The initial integer is: 491019
The value of compute_sum(491019) = 7
```

Note:

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

- 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 `iostream` (la libreria `<string>` è usato solo per conversione `char *` in intero).
- 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 funzione **ricorsiva** `compute_sum`, che prende come argomento un intero *positivo* `num` e restituisce:

- la somma delle cifre del numero se questa somma è minore di 10;
- il risultato della riapplicazione della funzione sulla somma calcolata addizionata di 10 se la somma calcolata è dispari;
- il risultato della riapplicazione della funzione sulla somma calcolata addizionata di 1 negli altri casi.

Per evitare ambiguità, riportiamo qui di seguito un esempio di applicazione delle regole qui sopra specificate. Sia `sum` una funzione che calcola la somma delle cifre che compongono il numero, `sum(99019) = 28` che è maggiore di 10 e pari, quindi sommo 1, `sum(29) = 11` che é maggiore di 10 e dispari, quindi sommo 10, `sum(21) = 3`, minore di 10 e quindi `compute_sum(99019) = 3`.

La funzione `compute_sum` **deve essere ricorsiva** e **NON deve contenere iteratori** espliciti (`for`, `while`, `do-while`). Sono **solo** consentite (se ritenute necessarie) chiamate a funzioni ricorsive ausiliarie che a loro volta **non** contengano iterazioni esplicite (`for`, `while`, `do-while`).

La funzione è inserita in un semplice programma che legge un intero positivo come unico argomento a linea di comando, lo converte in intero, chiama la funzione `compute_sum`, stampa a video il numero specificato ed il risultato della chiamata a `compute_sum`. Alcuni esempi di esecuzione sono i seguenti:

```
computer > ./a.out 38
The initial integer is: 38
The value of compute_sum(38) = 3
computer > ./a.out 99019
The initial integer is: 99019
The value of compute_sum(99019) = 3
computer > ./a.out 91019
The initial integer is: 91019
The value of compute_sum(91019) = 3
computer > ./a.out 191019
The initial integer is: 191019
The value of compute_sum(191019) = 4
computer > ./a.out 491011
The initial integer is: 491011
The value of compute_sum(491011) = 8
```

Note:

- Scaricare il file `esercizio2.cpp`, modificarlo per inserire la dichiarazione e la definizione della funzione `compute_sum`, e caricare il file sorgente risultato delle vostre modifiche a soluzione di questo esercizio nello spazio apposito.
- 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 `iostream` (la libreria `<string>` è usato solo per conversione `char *` in intero).

- 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 funzione **ricorsiva** `compute_sum`, che prende come argomento un intero *positivo* `num` e restituisce:

- la somma delle cifre del numero se questa somma è minore di 10;
- il risultato della riapplicazione della funzione sulla somma calcolata addizionata di 12 se la somma calcolata è multiplo di 3;
- il risultato della riapplicazione della funzione sulla somma calcolata addizionata di 3 negli altri casi.

Per evitare ambiguità, riportiamo qui di seguito un esempio di applicazione delle regole qui sopra specificate. Sia `sum` una funzione che calcola la somma delle cifre che compongono il numero, `sum(99019) = 28` che è maggiore di 10 e non è multiplo di 3, quindi sommo 3, `sum(31) = 4`, minore di 10 e quindi `compute_sum(99019) = 4`.

La funzione `compute_sum` **deve essere ricorsiva** e **NON deve contenere iteratori** espliciti (`for`, `while`, `do-while`). Sono **solo** consentite (se ritenute necessarie) chiamate a funzioni ricorsive ausiliarie che a loro volta **non** contengano iterazioni esplicite (`for`, `while`, `do-while`).

La funzione è inserita in un semplice programma che legge un intero positivo come unico argomento a linea di comando, lo converte in intero, chiama la funzione `compute_sum`, stampa a video il numero specificato ed il risultato della chiamata a `compute_sum`. Alcuni esempi di esecuzione sono i seguenti:

```
computer > ./a.out 38
The initial integer is: 38
The value of compute_sum(38) = 5
computer > ./a.out 99019
The initial integer is: 99019
The value of compute_sum(99019) = 4
computer > ./a.out 91019
The initial integer is: 91019
The value of compute_sum(91019) = 5
computer > ./a.out 191019
The initial integer is: 191019
The value of compute_sum(191019) = 6
computer > ./a.out 491019
The initial integer is: 491019
The value of compute_sum(491019) = 9
```

Note:

- Scaricare il file `esercizio2.cpp`, modificarlo per inserire la dichiarazione e la definizione della funzione `compute_sum`, e caricare il file sorgente risultato delle vostre modifiche a soluzione di questo esercizio nello spazio apposito.
- 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 `iostream` (la libreria `<string>` è usato solo per conversione `char *` in intero).

- 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 funzione **ricorsiva** `compute_sum`, che prende come argomento un intero *positivo* `num` e restituisce:

- la somma delle cifre del numero se questa somma è minore di 10;
- il risultato della riapplicazione della funzione sulla somma calcolata addizionata di 12 se la somma calcolata non è multiplo di 3;
- il risultato della riapplicazione della funzione sulla somma calcolata addizionata di 3 negli altri casi.

Per evitare ambiguità, riportiamo qui di seguito un esempio di applicazione delle regole qui sopra specificate. Sia `sum` una funzione che calcola la somma delle cifre che compongono il numero, `sum(99019) = 28` che è maggiore di 10 ed è multiplo di 3, quindi sommo 3, `sum(31) = 4`, minore di 10 e quindi `compute_sum(99019) = 4`.

La funzione `compute_sum` **deve essere ricorsiva** e **NON deve contenere iteratori** espliciti (`for`, `while`, `do-while`). Sono **solo** consentite (se ritenute necessarie) chiamate a funzioni ricorsive ausiliarie che a loro volta **non** contengano iterazioni esplicite (`for`, `while`, `do-while`).

La funzione è inserita in un semplice programma che legge un intero positivo come unico argomento a linea di comando, lo converte in intero, chiama la funzione `compute_sum`, stampa a video il numero specificato ed il risultato della chiamata a `compute_sum`. Alcuni esempi di esecuzione sono i seguenti:

```
computer > ./a.out 38
The initial integer is: 38
The value of compute_sum(38) = 5
computer > ./a.out 99019
The initial integer is: 99019
The value of compute_sum(99019) = 4
computer > ./a.out 91019
The initial integer is: 91019
The value of compute_sum(91019) = 5
computer > ./a.out 191019
The initial integer is: 191019
The value of compute_sum(191019) = 6
computer > ./a.out 491019
The initial integer is: 491019
The value of compute_sum(491019) = 9
```

Note:

- Scaricare il file `esercizio2.cpp`, modificarlo per inserire la dichiarazione e la definizione della funzione `compute_sum`, e caricare il file sorgente risultato delle vostre modifiche a soluzione di questo esercizio nello spazio apposito.
- 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 `iostream` (la libreria `<string>` è usato solo per conversione `char *` in intero).

- 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