Esame 20230619

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** `calcola_lista_medie`, che prende come primo argomento un albero binario di ricerca `t` di tipo `const tree *` di interi, come secondo argomento un puntatore **passato per riferimento** `r` di tipo `list *`.

La procedura `calcola_lista_medie` attraversa l'albero binario e per ogni ramo (percorso dalla radice ad un nodo con almeno un nodo `nullptr`) calcola la media dei soli nodi incontrati nel percorso che contengono nel campo `info` un numero pari.

La procedura `calcola_lista_medie` deve anche gestire i casi in cui l'albero binario di ricerca è vuoto, e/o non ci sono elementi pari.

La procedura `calcola_lista_medie` deve prima considerare il nodo sinistro, poi il nodo destro, e la ricorsione va prima a sinistra e poi a destra.

La procedura `calcola_lista_medie` **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 procedura `calcola_lista_medie` è inserita in un semplice programma che genera alcuni alberi binario di ricerca e per ognuno chiama la procedura stessa, stampa a video l'albero binario di ricerca, le liste costruite, e dealloca i diversi alberi e liste create.

Il `main` e le altre funzioni già presenti nel file `esercizio2.cpp` **NON DEVONO ESSERE MODIFICATE**. Un esempio di esecuzione é il seguente:

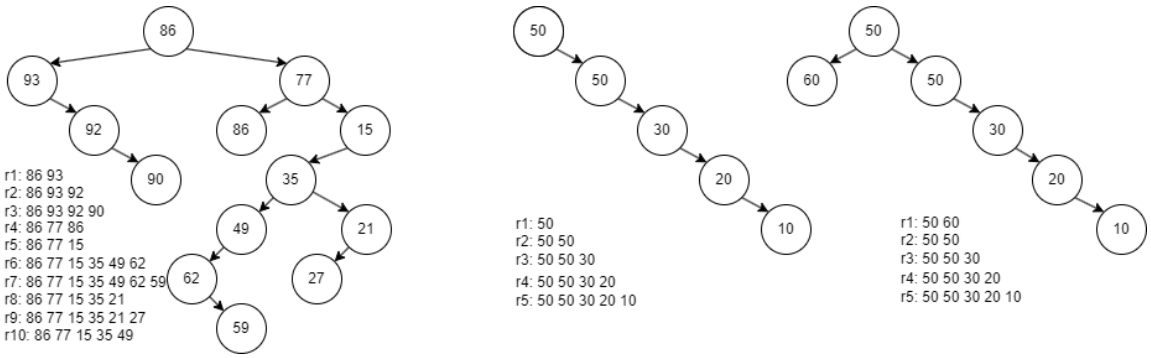
```
tree =
    15
        21
          27
      35
        49
            59
          62
  77
    86
86
       90
    92
  93

result = 86 86 74 74 86 86 86 89.3333 89 86
------------------------------------------------------------
tree =
        10
      20
    30
  50
50

result = 32 37.5 43.3333 50 50
------------------------------------------------------------
tree =
        10
      20
    30
```

```
  50
50
  60

result = 32 37.5 43.3333 50 55
------------------------------------------------------------
tree =

result =
```

Suggerimenti:

- Accumulare la somma dei valori che soddisfano la condizione, e il loro numero per poter calcolare le medie man mano che si attraversa l'albero.
- Ho un ramo quando
  - raggiungo un nodo per cui entrambi i campi `left` e `right` sono `nullptr`;
  - raggiungo un nodo per cui un ramo é `nullptr` mentre l'altro non é `nullptr`.

  La figura seguente mostra alcuni alberi (sono quelli del `main`) e i diversi rami (secondo la definizione di cui sopra).



Note:

- Scaricare il file `esercizio2.cpp`, modificarlo per inserire la dichiarazione e la definizione della funzione `calcola_lista_medie`, 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`, `cstdlib` e `ctime`.
- 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.
- Si ricorda di inserire solo nuovo codice e di **NON MODIFICARE** il resto del programma.

esercizio2.cpp

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** `calcola_lista_medie`, che prende come primo argomento un albero binario di ricerca `t` di tipo `const tree *` di interi, come secondo argomento un puntatore **passato per riferimento** `r` di tipo `list *`.

La procedura `calcola_lista_medie` attraversa l'albero binario e per ogni ramo (percorso dalla radice ad un nodo con almeno un nodo `nullptr`) calcola la media dei soli nodi incontrati nel percorso che contengono nel campo `info` un numero dispari.

La procedura `calcola_lista_medie` deve anche gestire i casi in cui l'albero binario di ricerca è vuoto, e/o non ci sono elementi dispari.

La procedura `calcola_lista_medie` deve prima considerare il nodo sinistro, poi il nodo destro, e la ricorsione va prima a sinistra e poi a destra.

La procedura `calcola_lista_medie` **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 procedura `calcola_lista_medie` è inserita in un semplice programma che genera alcuni alberi binario di ricerca e per ognuno chiama la procedura stessa, stampa a video l'albero binario di ricerca, le liste costruite, e dealloca i diversi alberi e liste create.

Il `main` e le altre funzioni già presenti nel file `esercizio2.cpp` **NON DEVONO ESSERE MODIFICATE**. Un esempio di esecuzione é il seguente:

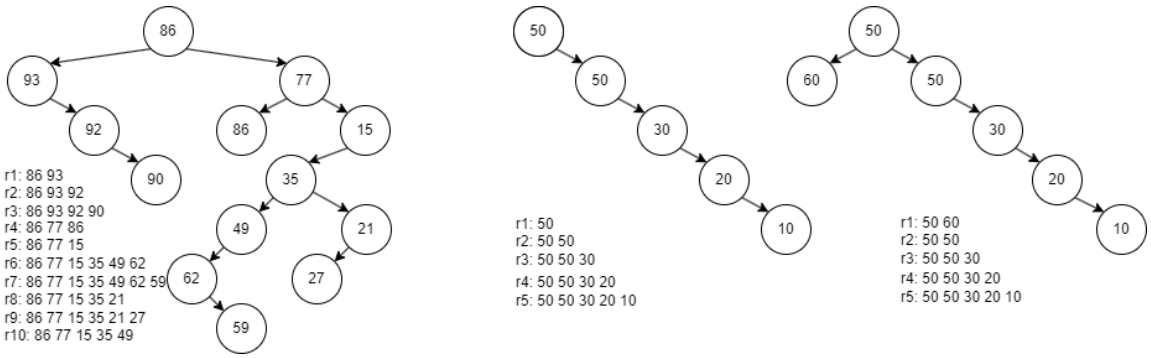
```
computer > ./esercizio2
tree =
    15
        21
          27
      35
        49
            59
          62
  77
    86
86
      90
    92
  93

result = 35 37 47 44 44 46 77 93 93 93
------------------------------------------------------------
tree =
        10
      20
    30
  50
50

result =
------------------------------------------------------------
tree =
        10
      20
    30
  50
50
  60
```

```
result =
------------------------------------------------------------
tree =

result =
```

Suggerimenti:

- Accumulare la somma dei valori che soddisfano la condizione, e il loro numero per poter calcolare le medie man mano che si attraversa l'albero.
- Ho un ramo quando
  - raggiungo un nodo per cui entrambi i campi `left` e `right` sono `nullptr`;
  - raggiungo un nodo per cui un ramo é `nullptr` mentre l'altro non é `nullptr`.

  La figura seguente mostra alcuni alberi (sono quelli del `main`) e i diversi rami (secondo la definizione di cui sopra).



Note:

- Scaricare il file `esercizio2.cpp`, modificarlo per inserire la dichiarazione e la definizione della funzione `calcola_lista_medie`, 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`, `cstdlib` e `ctime`.
- 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.
- Si ricorda di inserire solo nuovo codice e di **NON MODIFICARE** il resto del programma.

esercizio2.cpp

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** `calcola_lista_medie`, che prende come primo argomento un albero binario di ricerca `t` di tipo `const tree *` di interi, come secondo argomento un puntatore **passato per riferimento** `r` di tipo `list *`.

La procedura `calcola_lista_medie` attraversa l'albero binario e per ogni ramo (percorso dalla radice ad un nodo con almeno un nodo `nullptr`) calcola la media dei soli nodi incontrati nel percorso che contengono nel campo `info` un numero multiplo di 4.

La procedura `calcola_lista_medie` deve anche gestire i casi in cui l'albero binario di ricerca è vuoto, e/o non ci sono elementi multipli di 4.

La procedura `calcola_lista_medie` deve prima considerare il nodo sinistro, poi il nodo destro, e la ricorsione va prima a sinistra e poi a destra.

La procedura `calcola_lista_medie` **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 procedura `calcola_lista_medie` è inserita in un semplice programma che genera alcuni alberi binario di ricerca e per ognuno chiama la procedura stessa, stampa a video l'albero binario di ricerca, le liste costruite, e dealloca i diversi alberi e liste create.

Il `main` e le altre funzioni già presenti nel file `esercizio2.cpp` **NON DEVONO ESSERE MODIFICATE**. Un esempio di esecuzione é il seguente:

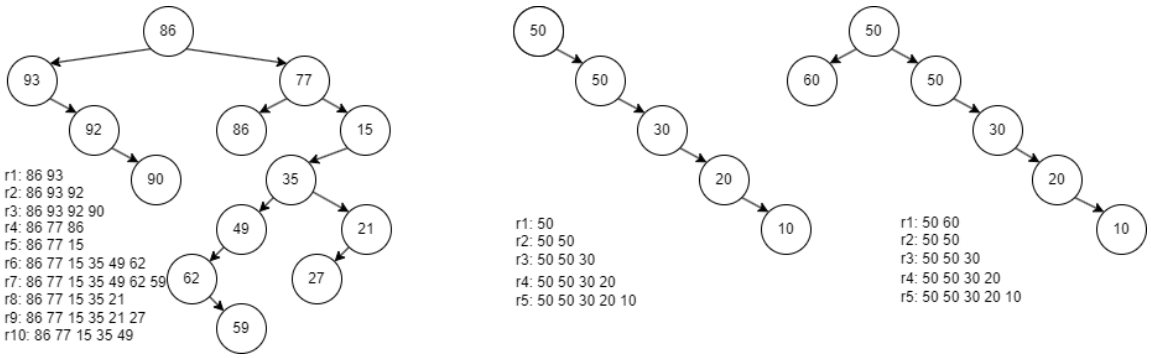
```
computer > ./esercizio2
tree =
    15
        21
          27
      35
        49
            59
          62
  77
    86
86
      90
    92
  93

result = 92 92
------------------------------------------------------------
tree =
        10
      20
    30
  50
50

result = 20 20
------------------------------------------------------------
tree =
        10
      20
    30
  50
50
  60
```

```
result = 20 20 60
-------------------------------------------------------
tree =

result =
```

Suggerimenti:

- Accumulare la somma dei valori che soddisfano la condizione, e il loro numero per poter calcolare le medie man mano che si attraversa l'albero.
- Ho un ramo quando
  - raggiungo un nodo per cui entrambi i campi `left` e `right` sono `nullptr`;
  - raggiungo un nodo per cui un ramo é `nullptr` mentre l'altro non é `nullptr`.

  La figura seguente mostra alcuni alberi (sono quelli del `main`) e i diversi rami (secondo la definizione di cui sopra).



Note:

- Scaricare il file `esercizio2.cpp`, modificarlo per inserire la dichiarazione e la definizione della funzione `calcola_lista_medie`, 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`, `cstdlib` e `ctime`.
- 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.
- Si ricorda di inserire solo nuovo codice e di **NON MODIFICARE** il resto del programma.

esercizio2.cpp

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** `calcola_lista_medie`, che prende come primo argomento un albero binario di ricerca `t` di tipo `const tree *` di interi, come secondo argomento un puntatore **passato per riferimento** `r` di tipo `list *`.

La procedura `calcola_lista_medie` attraversa l'albero binario e per ogni ramo (percorso dalla radice ad un nodo con almeno un nodo `nullptr`) calcola la media dei soli nodi incontrati nel percorso che contengono nel campo `info` un numero multiplo di 5.

La procedura `calcola_lista_medie` deve anche gestire i casi in cui l'albero binario di ricerca è vuoto, e/o non ci sono elementi multipli di 5.

La procedura `calcola_lista_medie` deve prima considerare il nodo sinistro, poi il nodo destro, e la ricorsione va prima a sinistra e poi a destra.

La procedura `calcola_lista_medie` **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 procedura `calcola_lista_medie` è inserita in un semplice programma che genera alcuni alberi binario di ricerca e per ognuno chiama la procedura stessa, stampa a video l'albero binario di ricerca, le liste costruite, e dealloca i diversi alberi e liste create.

Il `main` e le altre funzioni già presenti nel file `esercizio2.cpp` **NON DEVONO ESSERE MODIFICATE**. Un esempio di esecuzione é il seguente:

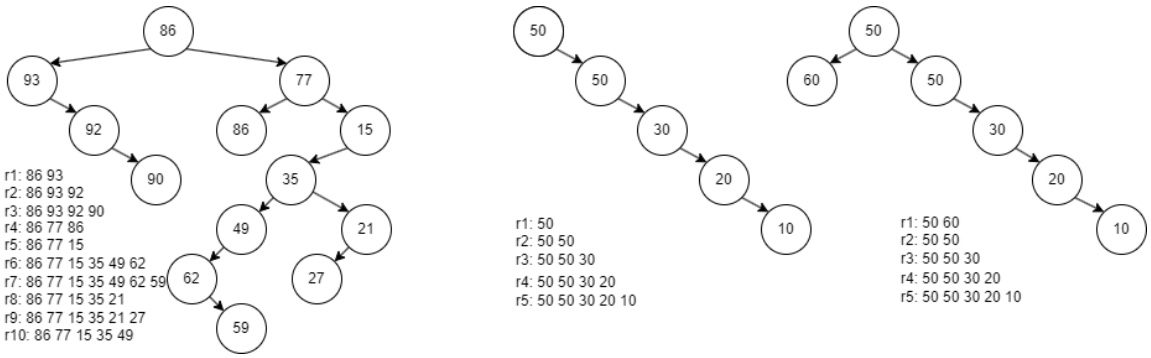
```
computer > ./esercizio2
tree =
    15
        21
          27
      35
        49
            59
          62
  77
    86
86
      90
    92
  93

result = 25 25 25 25 25 15 90
------------------------------------------------------------
tree =
        10
      20
    30
  50
50

result = 32 37.5 43.3333 50 50
------------------------------------------------------------
tree =
        10
      20
    30
  50
50
  60
```

```
result = 32 37.5 43.3333 50 55
------------------------------------------------------------
tree =

result =
```

Suggerimenti:

- Accumulare la somma dei valori che soddisfano la condizione, e il loro numero per poter calcolare le medie man mano che si attraversa l'albero.
- Ho un ramo quando
  - raggiungo un nodo per cui entrambi i campi `left` e `right` sono `nullptr`;
  - raggiungo un nodo per cui un ramo é `nullptr` mentre l'altro non é `nullptr`.

  La figura seguente mostra alcuni alberi (sono quelli del `main`) e i diversi rami (secondo la definizione di cui sopra).



Note:

- Scaricare il file `esercizio2.cpp`, modificarlo per inserire la dichiarazione e la definizione della funzione `calcola_lista_medie`, 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`, `cstdlib` e `ctime`.
- 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.
- Si ricorda di inserire solo nuovo codice e di **NON MODIFICARE** il resto del programma.

esercizio2.cpp

Information for graders:

Total of marks: 40