Esame 20230220

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** `estrai`, che prende come primo argomento un albero binario di ricerca `t` di tipo `const tree *` di interi, come secondo e quarto argomento due puntatori a interi `int * & l1`, `int * & l2` entrambi **passati per riferimento**, come terzo e quinto argomento due interi `l1_size` e `l2_size` entrambi **passati per riferimento**.

La procedura `estrai` estrae dall'albero binario di ricerca

- gli elementi **pari** e li memorizza nell'array `l1`;
- gli elementi **dispari** e li memorizza nell'array `l2`.

Gli array `l1` e `l2` devono essere allocati dinamicamente dalla procedura `estrai`, e devono contenere esattamente il numero di elementi necessari a contenere gli elementi da estrarre. La funzione `estrai` oltre ad allocare gli array `l1` e `l2` calcola la loro dimensione e ne ritorna il valore al chiamante attraverso i parametri `l1_size` e `l2_size`.

Gli elementi negli array `l1` e `l2` se stampati in ordine di indice dal più piccolo al più grande dovranno comparire in **ordine crescente**.

Gli array `l1` e `l2` dovranno contenere solo gli elementi estratti dall'albero binario di ricerca e non altri valori e/o valori ausiliari.

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

La procedura `estrai` **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 `estrai` è inserita in un semplice programma che genera un albero binario di ricerca `t` e chiama la procedura stessa, stampa a video l'albero binario di ricerca e gli array costruiti.

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
Initial tree content:  98 96 95 93 93 92 91 90 87 86 86 84 84 82 82 81 80
78 77 76 73 73 72 70 70 69 68 67 67 67 67 64 63 62 62 62 59 58 57 56 56 50
49 46 45 43 42 40 37 36 36 35 35 34 30 29 29 29 29 27 27 26 26 26 25 24 24
23 22 21 21 19 15 15 14 13 13 11 11 8 5 5 2
L1 = 2 8 14 22 24 24 26 26 26 30 34 36 36 40 42 46 50 56 56 58 62 62 62 64
68 70 70 72 76 78 80 82 82 84 84 86 86 90 92 96 98
L2 = 5 5 11 11 13 13 15 15 19 21 21 23 25 27 27 29 29 29 29 35 35 37 43 45
49 57 59 63 67 67 67 67 69 73 73 77 81 87 91 93 93 95
```

Suggerimenti:

- Calcolare attraverso una funzione di supporto le dimensioni degli array prima di procedere all'inserimento.
- Allocare gli array, e poi procedere all'inserimento degli elementi negli array.

Note:

- Scaricare il file `esercizio2.cpp`, modificarlo per inserire la dichiarazione e la definizione della funzione `estrai`, 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** `estrai`, che prende come primo argomento un albero binario di ricerca `t` di tipo `const tree *` di interi, come secondo e quarto argomento due puntatori a interi `int * & l1`, `int * & l2` entrambi **passati per riferimento**, come terzo e quinto argomento due interi `l1_size` e `l2_size` entrambi **passati per riferimento**.

La procedura `estrai` estrae dall'albero binario di ricerca

- gli elementi **pari** e li memorizza nell'array `l2`;
- gli elementi **dispari** e li memorizza nell'array `l1`.

Gli array `l1` e `l2` devono essere allocati dinamicamente dalla procedura `estrai`, e devono contenere esattamente il numero di elementi necessari a contenere gli elementi da estrarre. La funzione `estrai` oltre ad allocare gli array `l1` e `l2` calcola la loro dimensione e ne ritorna il valore al chiamante attraverso i parametri `l1_size` e `l2_size`.

Gli elementi negli array `l1` e `l2` se stampati in ordine di indice dal più piccolo al più grande dovranno comparire in **ordine decrescente**.

Gli array `l1` e `l2` dovranno contenere solo gli elementi estratti dall'albero binario di ricerca e non altri valori e/o valori ausiliari.

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

La procedura `estrai` **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 `estrai` è inserita in un semplice programma che genera un albero binario di ricerca `t` e chiama la procedura stessa, stampa a video l'albero binario di ricerca e gli array costruiti.

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
Initial tree content:  98 96 95 93 93 92 91 90 87 86 86 84 84 82 82 81 80
78 77 76 73 73 72 70 70 69 68 67 67 67 67 64 63 62 62 62 59 58 57 56 56 50
49 46 45 43 42 40 37 36 36 35 35 34 30 29 29 29 29 27 27 26 26 26 25 24 24
23 22 21 21 19 15 15 14 13 13 11 11 8 5 5 2
L1 = 95 93 93 91 87 81 77 73 73 69 67 67 67 67 63 59 57 49 45 43 37 35 35
29 29 29 29 27 27 25 23 21 21 19 15 15 13 13 11 11 5 5
L2 = 98 96 92 90 86 86 84 84 82 82 80 78 76 72 70 70 68 64 62 62 62 58 56
56 50 46 42 40 36 36 34 30 26 26 26 24 24 22 14 8 2
```

Suggerimenti:

- Calcolare attraverso una funzione di supporto le dimensioni degli array prima di procedere all'inserimento.
- Allocare gli array, e poi procedere all'inserimento degli elementi negli array.

Note:

- Scaricare il file `esercizio2.cpp`, modificarlo per inserire la dichiarazione e la definizione della funzione `estrai`, 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** `estrai`, che prende come primo argomento un albero binario di ricerca `t` di tipo `const tree *` di interi, come secondo e quarto argomento due puntatori a interi `int * & l1`, `int * & l2` entrambi **passati per riferimento**, come terzo e quinto argomento due interi `l1_size` e `l2_size` entrambi **passati per riferimento**.

La procedura `estrai` estrae dall'albero binario di ricerca

- gli elementi **pari** e li memorizza nell'array `l1`;
- gli elementi **dispari** e li memorizza nell'array `l2`.

Gli array `l1` e `l2` devono essere allocati dinamicamente dalla procedura `estrai`, e devono contenere esattamente il numero di elementi necessari a contenere gli elementi da estrarre. La funzione `estrai` oltre ad allocare gli array `l1` e `l2` calcola la loro dimensione e ne ritorna il valore al chiamante attraverso i parametri `l1_size` e `l2_size`.

Gli elementi negli array `l1` e `l2` se stampati in ordine di indice dal più piccolo al più grande dovranno comparire in **ordine decrescente**.

Gli array `l1` e `l2` dovranno contenere solo gli elementi estratti dall'albero binario di ricerca e non altri valori e/o valori ausiliari.

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

La procedura `estrai` **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 `estrai` è inserita in un semplice programma che genera un albero binario di ricerca `t` e chiama la procedura stessa, stampa a video l'albero binario di ricerca e gli array costruiti.

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
Initial tree content:  98 96 95 93 93 92 91 90 87 86 86 84 84 82 82 81 80
78 77 76 73 73 72 70 70 69 68 67 67 67 67 64 63 62 62 62 59 58 57 56 56 50
49 46 45 43 42 40 37 36 36 35 35 34 30 29 29 29 29 27 27 26 26 26 25 24 24
23 22 21 21 19 15 15 14 13 13 11 11 8 5 5 2
L1 = 98 96 92 90 86 86 84 84 82 82 80 78 76 72 70 70 68 64 62 62 62 58 56
56 50 46 42 40 36 36 34 30 26 26 26 24 24 22 14 8 2
L2 = 95 93 93 91 87 81 77 73 73 69 67 67 67 67 63 59 57 49 45 43 37 35 35
29 29 29 29 27 27 25 23 21 21 19 15 15 13 13 11 11 5 5
```

Suggerimenti:

- Calcolare attraverso una funzione di supporto le dimensioni degli array prima di procedere all'inserimento.
- Allocare gli array, e poi procedere all'inserimento degli elementi negli array.

Note:

- Scaricare il file `esercizio2.cpp`, modificarlo per inserire la dichiarazione e la definizione della funzione `estrai`, 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** `estrai`, che prende come primo argomento un albero binario di ricerca `t` di tipo `const tree *` di interi, come secondo e quarto argomento due puntatori a interi `int * & l1`, `int * & l2` entrambi **passati per riferimento**, come terzo e quinto argomento due interi `l1_size` e `l2_size` entrambi **passati per riferimento**.

La procedura `estrai` estrae dall'albero binario di ricerca

- gli elementi **pari** e li memorizza nell'array `l2`;
- gli elementi **dispari** e li memorizza nell'array `l1`.

Gli array `l1` e `l2` devono essere allocati dinamicamente dalla procedura `estrai`, e devono contenere esattamente il numero di elementi necessari a contenere gli elementi da estrarre. La funzione `estrai` oltre ad allocare gli array `l1` e `l2` calcola la loro dimensione e ne ritorna il valore al chiamante attraverso i parametri `l1_size` e `l2_size`.

Gli elementi negli array `l1` e `l2` se stampati in ordine di indice dal più piccolo al più grande dovranno comparire in **ordine crescente**.

Gli array `l1` e `l2` dovranno contenere solo gli elementi estratti dall'albero binario di ricerca e non altri valori e/o valori ausiliari.

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

La procedura `estrai` **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 `estrai` è inserita in un semplice programma che genera un albero binario di ricerca `t` e chiama la procedura stessa, stampa a video l'albero binario di ricerca e gli array costruiti.

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
Initial tree content:  98 96 95 93 93 92 91 90 87 86 86 84 84 82 82 81 80
78 77 76 73 73 72 70 70 69 68 67 67 67 67 64 63 62 62 62 59 58 57 56 56 50
49 46 45 43 42 40 37 36 36 35 35 34 30 29 29 29 29 27 27 26 26 26 25 24 24
23 22 21 21 19 15 15 14 13 13 11 11 8 5 5 2
L1 = 5 5 11 11 13 13 15 15 19 21 21 23 25 27 27 29 29 29 29 35 35 37 43 45
49 57 59 63 67 67 67 67 69 73 73 77 81 87 91 93 93 95
L2 = 2 8 14 22 24 24 26 26 26 30 34 36 36 40 42 46 50 56 56 58 62 62 62 64
68 70 70 72 76 78 80 82 82 84 84 86 86 90 92 96 98
```

Suggerimenti:

- Calcolare attraverso una funzione di supporto le dimensioni degli array prima di procedere all'inserimento.
- Allocare gli array, e poi procedere all'inserimento degli elementi negli array.

Note:

- Scaricare il file `esercizio2.cpp`, modificarlo per inserire la dichiarazione e la definizione della funzione `estrai`, 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