Esercizio 11 – Alberi

Implementare il proprio tipo di dato Albero Binario con le seguenti funzioni:

newBtree : *( ) → ALBEROBIN*

emptyBtree : *(ALBEROBIN) → BOOLEAN*

getRoot : *(ALBEROBIN) → NODO*

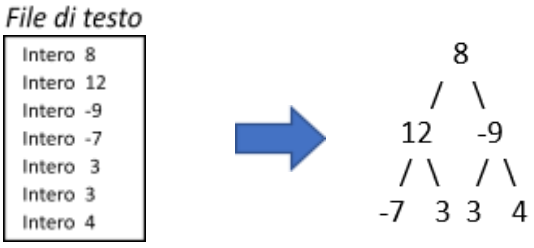
figlioSX: *(ALBEROBIN) → ALBEROBIN*

figlioDX : *(ALBEROBIN) → ALBEROBIN*

consBtree : *(ITEM, ALBEROBIN, ALBEROBIN) → ALBEROBIN*

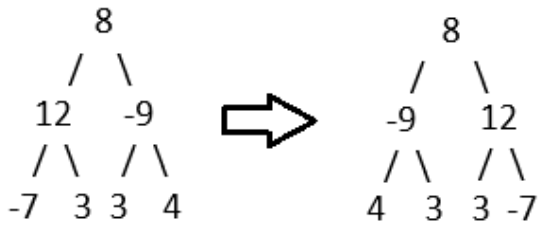
Dopodiché scrivere il codice ai seguenti problemi:

1. **Creazione da file**: Con la funzione inputBTree, realizzare un albero binario i cui nodi contengono dei numeri interi letti da file. Dato quindi un file di testo che contiene dei numeri interi, la lettura del file deve essere tale che l'albero risultante sia il seguente:



2. **Conta foglie**: Un nodo foglia è un nodo che non ha ulteriori ramificazioni. Realizzare una funzione che conti il numero di foglie presenti nell'albero.

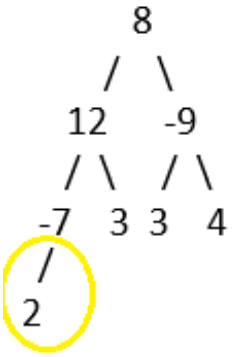
3. **Albero speculare:** l'albero speculare è quello ottenuto invertendo ciascun nodo sinistro con il nodo destro. Realizzare una funzione che dato un albero binario ***crei*** l'albero speculare `BTree speculare(BTree albero).`



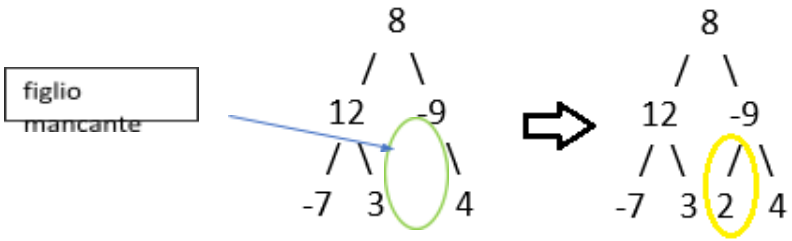
4. **Min/Max**: Scrivere due funzioni di ricerca del nodo con valore minimo e massimo, rispettivamente, e restituire tale valore in output. Si è liberi di scegliere la visita dell'albero che si preferisce, dopotutto vanno visitati tutti i nodi dell'albero binario per poter individuare il nodo con valore minimo e massimo.

5. **Uguaglianza tra alberi**: due alberi sono uguali se contengono lo stesso numero di nodi disposti nella stessa maniera e con gli stessi valori. Implementare una funzione `char uguali (BTree albero1, BTree albero2)` che dati in input due alberi dica se essi sono uguali.
   (Opzionale: se i due alberi non sono uguali, la funzione potrebbe restituire le informazioni (valore/livello/sx-dx) del primo nodo a differire tra albero 1 e albero2).

6. **Aggiungi nodo:** realizzare una funzione `aggiungiNodo(BTree T, item nodo)` che inserisca nella prima posizione "utile" dell'albero T il nodo dato in input. Più precisamente, la prima posizione utile è il primo "figlio mancante" (sinistro o destro) che si incontra durante una visita per livelli dell'albero.
   Nell'esempio considerato: se `nodo=2` il risultato sarebbe il seguente:



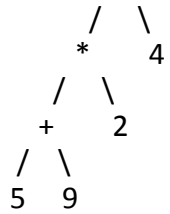
   Ma se l'albero fosse diverso come qui di seguito, l'inserimento avverrebbe diversamente come mostrato:



7. **[*avanzato*]Valuta espressione**: un albero può essere una struttura ideale a rappresentazione delle espressioni nella notazione infissa. I nodi dell'albero costituiscono le operazioni mentre le foglie rappresentano gli operandi. Realizzare un programma che attraverso una visita dell'albero calcoli il risultato dell'espressione da esso rappresentato. La funzione prende in input un albero che si suppone codifichi correttamente l'espressione che si vuole valutare.

   `float valuta(Btree T)`

   Esempio:
   Data la seguente espressione (5 + 9) * 2 - 4
   Ne risulterà il seguente albero binario da dare in input alla funzione:



Il risultato restituito dalla funzione sarà: 24

Approfondimento: chi vuole, può provare anche realizzare la funzione che costruisce l'albero data la stringa dell'espressione.
Prestare attenzione al fatto che la presenza delle parentesi ha un impatto sulla struttura dell'albero risultante. *In assenza di parentesi i più volenterosi potrebbero addirittura porsi il problema di controllare la precedenza degli operatori.*