**Attività 12 – Alberi di ricerca**

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

*BST newBST(void);*

*int emptyBST(BST T);*

*BST figlioSX(BST T);*

*BST figlioDX(BST T);*

*BST insert(BST T, item elem);*

*int contains(BST T, item elem);*

*BST deleteNode(BST T, item elem);*

*item root(BST T);*

Dopodiché scrivere il codice ai seguenti problemi:

1. **E’ di ricerca?**: Dato un albero binario BTree, scrivere una funzione che attraverso la restituzione di un valore booleano verifichi che esso sia un albero binario di ricerca.   
   bool isBST(BTree tree)
2. **Nodi in un Intervallo**: Dato un albero binario di ricerca T i cui valori sono interi e due valori interi A e B (tale che A<B), stampare in ordine crescente tutti i nodi di T che sono compresi nell’intervallo [A, B] (estremi inclusi).
3. **Bilanciato**: Dato un array di interi in input, costruire un albero binario di ricerca perfettamente bilanciato i cui nodi siano popolati dagli elementi dell’array.  
   NOTA. *l’array in input che caratteristiche dovrebbe avere?  
   La funzione insert non deve già prevedere l’implementazione delle rotazioni.*
4. **Nodi ad altezza k**: Dato un albero binario di ricerca T i cui item sono interi e un intero k, scrivere una funzione che restituisca tutti i nodi di T che hanno altezza k. L’output è una struttura dati a propria scelta.
5. **Mediano:** Dato un albero binario di ricerca T i cui item sono numeri interi, e un valore intero U, stabilire se il nodo il cui valore è pari ad U in T è il nodo il cui valore è il mediano del sottoalbero di cui esso è radice.  
   Nota: data una distribuzione ordinabile, si definisce mediana il valore assunto dalla unità statistica che si trova nel mezzo della distribuzione.

| 2 | 5 | 8 | 88 | 102 | 145 | 189 |
| --- | --- | --- | --- | --- | --- | --- |

Mediano

1. **Numero archi tra nodi[avanzato]**: dato un albero binario di ricerca T i cui item sono numeri interi positivi, e dati due valori X e Y (con X < Y), verificare che esistano i nodi nell’albero i cui valori sono corrispondenti a X e Y e calcolare il numero di archi che portano da X a Y.  
   *Suggerimento: i due nodi si troveranno sempre nel sottoalbero radicato in un nodo con valore Z tale che X<=Z<=Y*
2. **Antenato comune[avanzato]**: dato un albero binario di ricerca T i cui item sono interi e il valore di due item interi A e B, trovare il loro antenato comune più prossimo (*lowest common ancestor, LCA*).  
   NOTA. La radice è l’antenato comune più prossimo dei suoi nodi figli. Due nodi non-fratelli avranno necessariamente un antenato comune che differisce dai loro rispettivi nodi padre.

