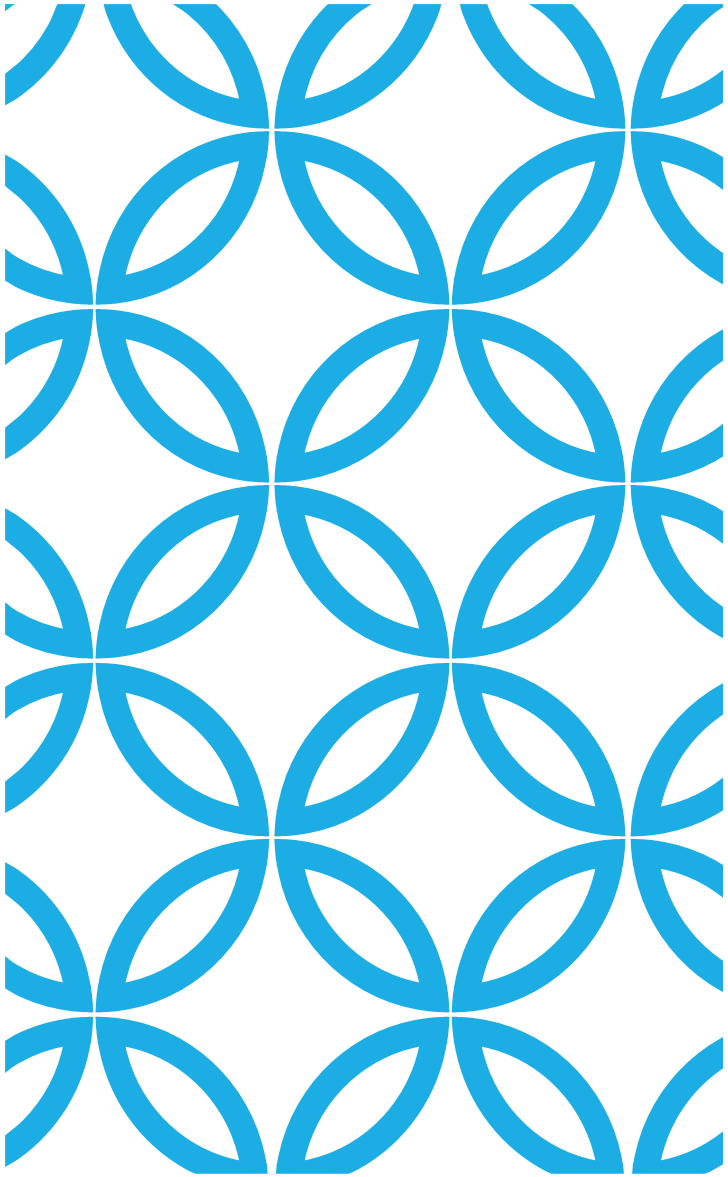

ADT ALBERO BINARIO 2

Corso di Programmazione e Strutture Dati
Docente di Laboratorio: Marco Romano
Email: marromano@unisa.it

ESERCIZI

1. Scrivere una funzione che cerchi un intero **k** all'interno di un albero binario.
2. Scrivere una funzione che restituisca il valore massimo contenuto nei nodi di un albero binario.
3. Scrivere una funzione che verifichi se due alberi binari sono uguali

(**Scrivere sempre pre e post condizione di ogni funzione)**

ESERCIZIO 1

Scrivere una funzione che cerchi un intero k all'interno di un albero binario.

Pre condizioni:

- La funzione prende in ingresso un albero binario e un intero k da cercare nei nodi dell'albero

Post condizioni:

- La funzione restituisce 1 se k è contenuto in almeno un nodo, 0 altrimenti. Se l'albero è vuoto, la funzione restituisce 0.

```
179   int findNumber(BTree T, int k) {
180     if (isEmptyTree(T))
181         return 0;
182
183     int *pt = T->value;
184     return *pt == k
185             || findNumber(getLeft(T),k)
186             || findNumber(getRight(T),k);
187   }//chiude findNumber
```

ESERCIZIO 2

Scrivere una funzione che restituisca il valore massimo contenuto nei nodi di un albero.

Pre condizioni:

- la funzione prende in ingresso un albero binario

Post condizioni:

- La funzione restituisce il valore massimo tra quelli presenti nei nodi dell'albero, -1 se l'albero è vuoto.

```
190   int max3(int a, int b, int c){
191       return a > b ? (c > a ? c : a) : (c > b ? c : b);
192   }//chiude Max3
193   int getMax(BTree T) {
194       if (isEmptyTree(T))
195         return -1;
196       int max_dx, max_sx;
197       int *pt = T->value;
198
199       max_sx=getMax(getLeft(T));
200       max_dx=getMax(getRight(T));
201
202       return max3(max_sx, max_dx, *pt);
203     }//chiude getMax
```

ESERCIZIO 3

Scrivere una funzione che verifichi se due alberi binari sono uguali.

Pre condizioni:

- la funzione prende in ingresso due alberi binari eventualmente vuoti

Post condizioni:

- la funzione restituisce 1 se i due alberi hanno la stessa struttura e i medesimi contenuti

```
205     int uguali(BTree T1, BTree T2) {
206       if ( isEmptyTree(T1) && isEmptyTree(T2) ) return 1;
207       if ( isEmptyTree(T1) || isEmptyTree(T2) ) return 0;
208       int *pt1 = T1->value;
209       int *pt2 = T2->value;
210
211       return *pt1==*pt2
212               && uguali(getLeft(T1), getLeft(T2))
213               && uguali(getRight(T1), getRight(T2));
214     }//chiude uguali
```

ESERCIZIO PER CASA

Sviluppare la funzione **mergesort** in forma iterativa

Suggerimento: potete riutilizzare la funzione **merge**