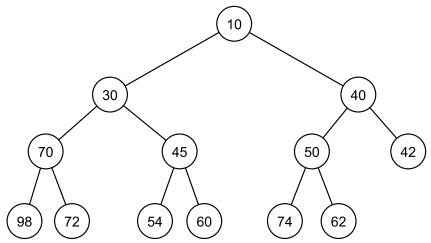

Aufgabe 11.1
Implementieren Sie das Interface PrioSchlange<T **extends** Comparable<T>> in einer Klasse MinHeap<T>. Verwenden Sie als interne Datenstruktur ein dynamisches Array.

Aufgabe 11.2 (Theorie)
Betrachten Sie folgende (Baum-)Darstellung eine MinHeaps:



Geben Sie an (ebenfalls in Baum-Darstellung), wie sich der Heap durch die folgenden Operationen verändert:

removeMin(), insert (47), insert (11), removeMin()

Bedenken Sie, dass nach jeder insert- bzw remove-Operation auch eine der „reHeap"-Operationen upheap() bzw downheap() erforderlich ist. Zeichnen Sie den Baum nach jeder Operation neu!

Aufgabe 11.3 (Theorie)
Lösen Sie Aufgabe 11.2 erneut, aber geben Sie diesmal die Veränderungen in *Array-Darstellung* an.

Aufgabe 11.4 (Theorie)
Benutzen Sie das Bottom-Up-HeapSort-Verfahren (mit einem MaxHeap), (nicht um das folgende Array komplett zu sortieren, sondern) um **die drei größten Elemente** des folgenden Arrays zu bestimmen:

48 63 5 24 11 84 50 12 97 28

a) Bauen Sie nach der Bottom-Up-Methode aus dem Array einen MaxHeap in **Array-Darstellung** auf. Geben Sie jeweils an, für welches Element Sie den *downheap()* durchführen.
Geben Sie nach jeder Vertauschung die getauschten Werte an. Gleichbleibende Elemente brauchen Sie **während eines downheap()-Durchlaufs** nicht abschreiben - Sie sollten aber zur eigenen besseren Orientierung **nach jedem** Durchlauf das Array komplett angeben.

b) Führen Sie nun - **wieder in Array-Darstellung** - dreimal die Operation *removeMax()* mit den erforderlichen *downheap()*-Operationen aus (auch nach der dritten *removeMax()*-Operation).

Achtung!
In der Vorlesung wurden die up- und downheap-Operationen nur für **Min**Heaps formuliert. Was muss bei einem **Max**Heap geändert werden?