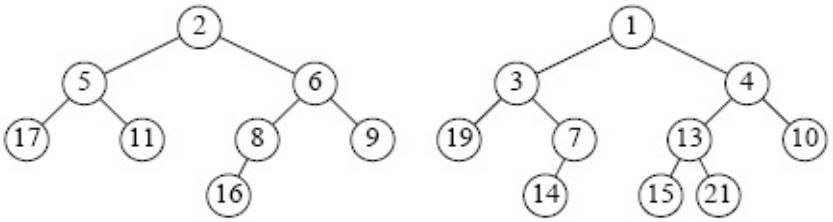
NAME: ..... **Student ID.**: .....

1. (10 pts) Consider the two heaps shown in the figure below. Show the result of merging these two heaps using the merge algorithm for (a) *leftist heaps*, and (b) *skew heaps*.

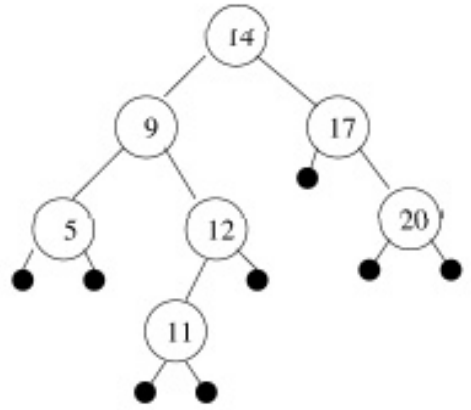


2. (15 pts) Consider the following tree.

   (a) (5 pts) Label each node in the tree with r or b denoting the colors RED and BLACK, respectively, so that the tree is a legal red-black tree.

   (b) (5 pts) In the red-black tree you just constructed in (a), perform an insertion of 10. Show the resulting tree (with nodes labelled with r or b).

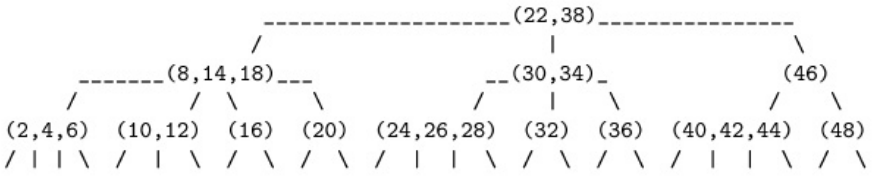
   (c) (5 pts) In the red-black tree you just constructed in (b), perform a deletion of 5. Show the resulting tree (with nodes labelled with r or$b$).



3. (10 pts) Consider the following 2-3-4 tree.

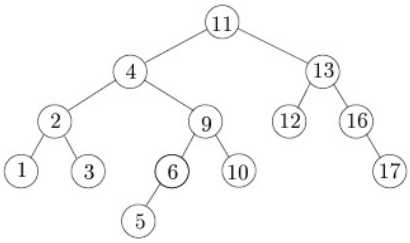
   (a) (5 pts) Delete 16 from the tree using a bottom-up deletion approach

   (b) (5 pts) Insert 11 into the original tree using a top-down insertion approach

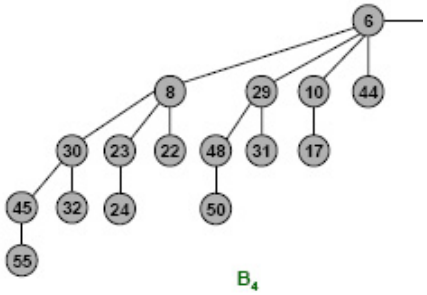


4. (10 pts) Suppose we have an array $A[1, ..., m]$ supporting the following two operations (initially, $k = 0$).
   $Insert(x)$: $k := k + 1$; $A[k] := x$
   $Half()$: for $i = 1$ to k do print $A[i]$ end-for; $k := \lfloor \frac{k}{2} \rfloor$

   Suppose we start with an empty array $A[...]$. Clearly $Insert(x)$ takes $\Theta(1)$ time and $Half()$ takes $\Theta(k)$ time. Show that the amortized complexity per operation is $\Theta(1)$. (Hint: Use the potential method)



6. (10 pts) The following figure shows part of a Fibonacci tree. We know that decrease-key in Fibonacci heaps is carried out by "cutting" and "marking". Suppose we want to cut as many nodes in the tree as possible without causing any "cascading promotions" (i.e., removing a node triggers the cutting of its parent node). What is the resulting tree? Be sure to clearly specify marked/unmarked nodes in the final tree.



7. (10 pts) True or false? Mark 'O' for true; '$\times$' for false. No penalty for wrong answer.

   (a) ..... Given an unsorted array of n elements, it is always possible to construct a min-binary-heap from the array in $O(n)$ time.

   (b) ..... Given a min-binary-heap, the maximum element can be found in $O(logn)$ time.

   (c) ..... $\frac{logn}{n} < log^* n < logn$.

   (d) ..... The length of the longest path in a leftist tree of n keys is bounded by $O(log\ n)$.

   (e) ..... The expected number of levels in a skip list of n keys is bounded by $O(log\ n)$.

   (f) ..... Double hashing is a hash table design that combines both chaining and open addressing.

   (g) ..... The maximum degree $D(n)$ of any node in an n-node Fibonacci heap is bounded by $O(logn)$.

   (h) ..... A decrease key operation in a binomial heap of n keys may take $O(n)$ time in the worst case.

   (i) ..... Let $B(n)$ be the number of nodes in the n-th binomial tree. We have $B(n) = 2 \times B(n-1) + 1$.

   (j) ..... In hashing, collisions will never occur if we use the *chaining* approach.

8. (15 pts) Give the asymptotic bound that applies to each of the quantities below. Choose your answer from this list: $\Theta(1), \Theta(log\ n), \Theta(log^2 n), \Theta(n), \Theta(n\ log\ n), \Theta(n^2), \Theta(n^2 log\ n), \Theta(n^2 log^2 n), \Theta(log^* n), \Theta(n\ log^* n)$. No penalty for wrong answer. (Heaps are assumed to be min-heaps.)

   (a) ..... The worst-case time to build a binomial heap with n keys.

   (b) ..... The worst-case time to build an AVL tree with n keys.

   (c) ..... The worst-case time to search an arbitrary key in a binary-heap with n^2 keys.

   (d) ..... The height of a red-blank tree with 2^n keys.

   (e) ..... The worst-case time to find a key in a skip list of n keys.

   (f) ..... The worst-case time to union two binomial heaps, each having n nodes.

   (g) ..... The worst-case time to find the minimum (without deleting it) of a binomial-heap of n^2 keys.

   (h) ..... The amortized time of a decrease-key operation in a Fibonacci-heap of n keys.

   (i) ..... The worst-case time of a delete-minimum operation in a Fibonacci-heap of n keys.

   (j) ..... The worst-case time of a UNION in *array* implementation of UNION-FIND with n elements.

9. (10 pts) Show the resulting tree after inserting the following key (in the given order) into an initially empty AA-tree: 5, 6, 7, 8, 9, 10, 1, 2, 3, 4 . Show your derivation in sufficient detail.