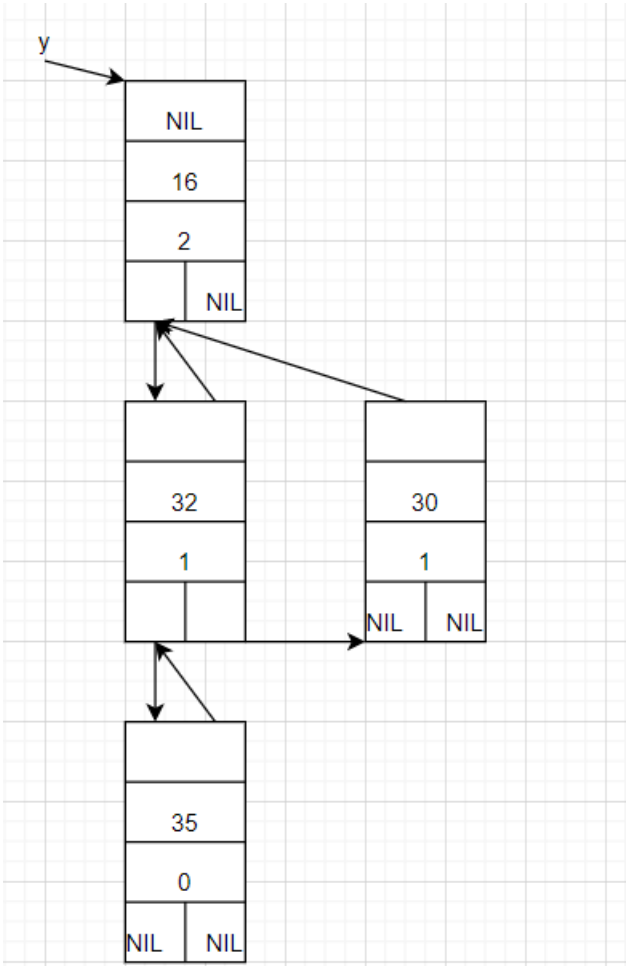
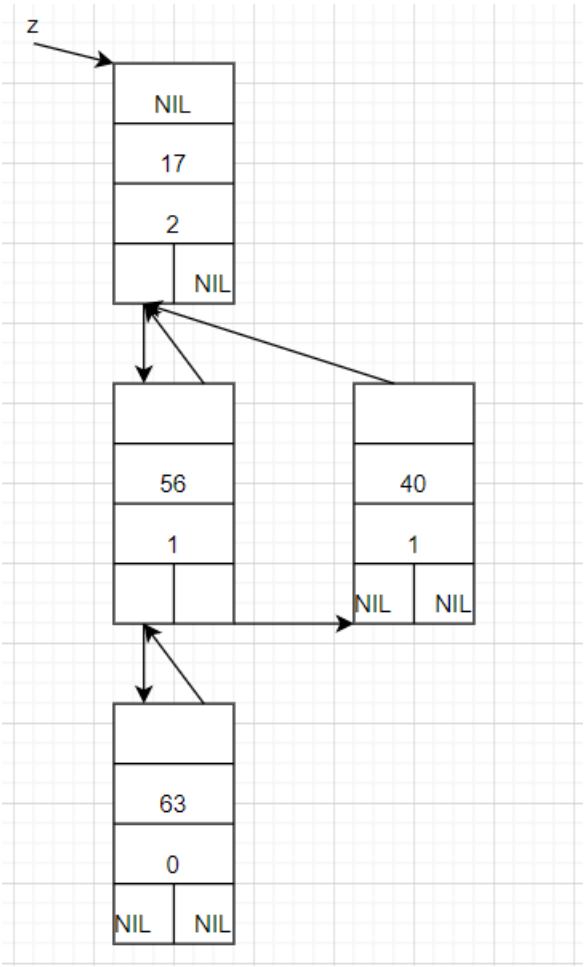
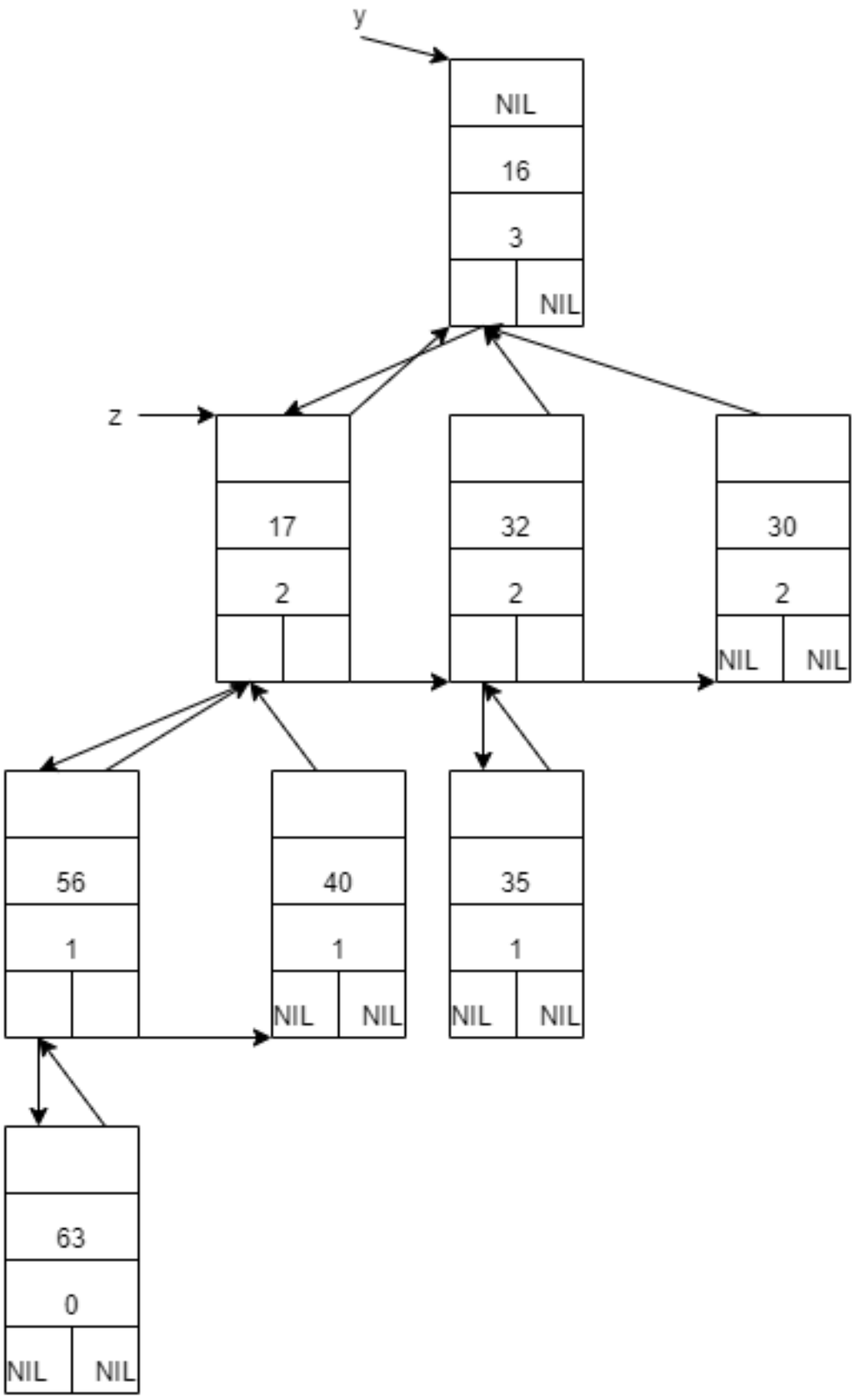
I certify that every answer in this assignment is the result of my own work; that I have neither copied off the Internet nor from any one else's work; and I have not shared my answers or attempts at answers with anyone else.

1: Consider the two B2 trees shown below. Show the data structures (based on what we had defined in class) corresponding to the trees that y and z point to. Now show the resulting data structure after an invocation of BINOMIALLINK(z,y).

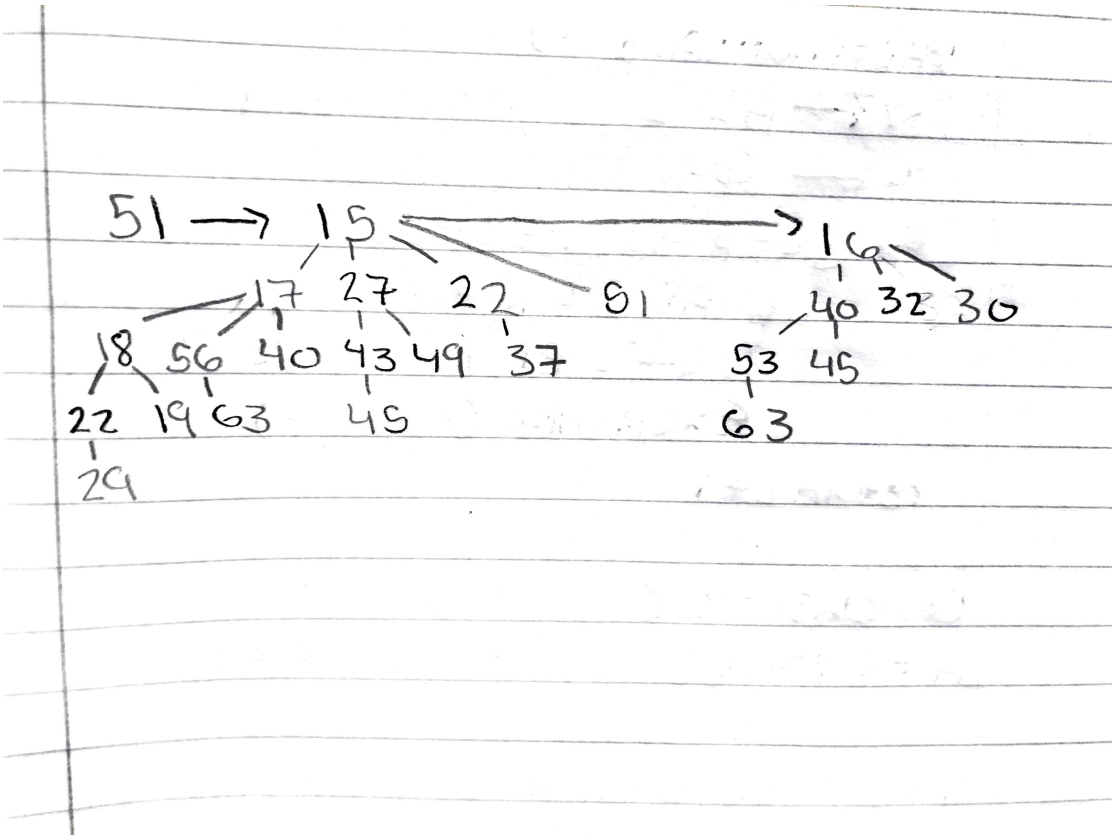




After BINOMIALLINK(z,y)


y
NIL
16
3
NIL
z
17
2
32
2
30
2
NIL
NIL
56
1
40
1
NIL
NIL
35
1
NIL
NIL
63
0
NIL
NIL

2: (Using the algorithms discussed in class, show the binomial heap that results when BINOMIALHEAPEXTRACTMIN is invoked on the following. It is enough to show the final heap.(You may insert a scanned file of your hand-drawn figure. However, it must be absolutely clear.)



3: Consider the QUICKSORT we have seen in class.Suppose the initial invocation is QUICKSORT(A, 1, 9), where A[1..9] =<27, 55, 2, 48, 96, 19, 41, 2, 27>.•What is the value of q returned by the very first call to PARTITION?•What are the subarrays of A in the two recursive calls to QUICKSORT immediately thereafter?•Draw the entire recursion tree generated from that initial invocation. Follow our notation: each node containing the array segment size inside it and annotated with the non-recursive time outside it.

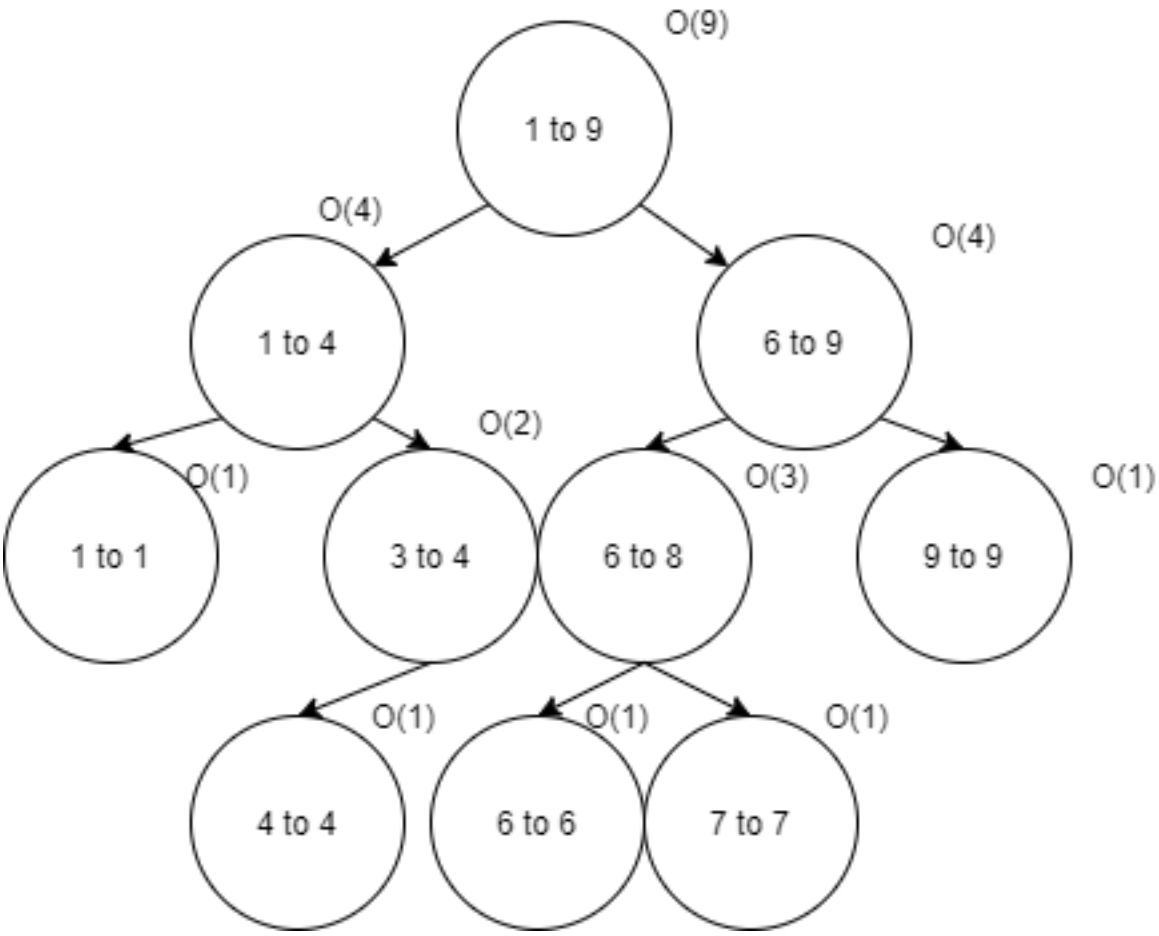
These were the functions provided to us:

```
QUICKSORT(A, p, r)
1  if p < r then
2      q ← PARTITION(A, p, r)
3      QUICKSORT(A, p, q − 1)
4      QUICKSORT(A, q + 1, r)
```

```
PARTITION(A, p, r)
1  v ← A[r] ▷ v is the pivot; i indexes the last
2  i ← p − 1 ▷ element in the left segment and
3  ▷ j just left of the start of the right segment
4  for j ← p to (r − 1) do
5      if A[j] ≤ v then ▷ A[j] in wrong segment!
6          i ← i + 1 ▷ So, extend the left segment
7          EXCHANGE(A, i, j)
8  EXCHANGE(A, i + 1, r) ▷ install the pivot
9  return i + 1 ▷ return the index of the pivot
```

QUICKSORT(A,1,9) would result in the first call to partition being
q = PARTITION(A,1,9)
in Partition:
PARITION(A, 1, 9)
v = A[9] = 27
i = 1-1 = 0
for j = 1 to 8
1: true, i = 1, Exchange(A,2,9), <27,27,2,48,96,19,41,2,55>
2:true, i = 2, Exchange(A,3,9), <27,27,55,48,96,19,41,2,2>
3: false
4: false
5 false
6: true, i =3, Exchange(A,4,9), <27,27,55,2,96,19,41,2,48>
7: false
8: true, i = 4, Exchange(A,5,9), <27,27,55,2,48,19,41,2,96>
return 5
q = 5

That means subarrays are:
A[p to q-1] = A[1 to 4] = $< 27, 27, 25, 2 >$
A[q+1 to r] = A[6 to 9] = $< 19, 41, 2, 96 >$


O(9)
1 to 9
O(4)
1 to 4
O(4)
6 to 9
O(1)
1 to 1
O(2)
3 to 4
O(3)
6 to 8
O(1)
9 to 9
O(1)
4 to 4
O(1)
6 to 6
O(1)
7 to 7