Tutorial #4

Question 1a

Max number of keys per node = \( m-1 = 5-1 = 4 \)

Min number of keys per node = \( \frac{m}{2}-1 = 3-1 = 2 \)

Inserting Z into node A, results in split and middle key X passed to node B
Question 1a

Node C has too many keys so it splits and middle key P is passed up a level resulting in creation of new root.

Question 1b

Deleting U, replaces it with V, its lexicographic successor.

Node A now has too few keys so it merges with left hand brother node B and common parent node V.
Question 1b

Node D has too many keys and node E has too few, so node D splits and passes up middle key T to node E. Deletion of U is now complete.

Question 2

Max number of keys per node = m - 1 = 5 - 1 = 4
Min number of keys per node = (m/2) - 1 = 3 - 1 = 2

Keys 13 17 inserted into node A but 19 causes it to split and 13 passed up to B.
Keys 23, 29 are inserted into A but 31 causes A to split, 23 is passed up to B for insertion.

Keys 37, 41 are inserted into A but 43 causes node to split and 37 is passed up to node B for insertion.
Keys 47, 53 are inserted into node A but key 59 causes a split; key 47 is passed up to node B which in turn has too many keys. Node B splits and 23 is passed up a level to create a new root.

**Question 2a&2b**

a)  
- Total no of keys = 17
- Total no of splits = 6
- Average number of node splits per key inserted = \(6/17\)

b)  
- Average search time = total search time/number of keys  
  = \((1 + 4*2 + 12*3)/17\)  
  = \(45/17\)
Question 2c

Optimal **binary** tree on the same keyset, 17 keys
- 1 @ level 1
- 2 @ level 2
- 4 @ level 3
- 8 @ level 4
- 2 @ level 5

Average search time is
- \((1 + 2 \times 2 + 3 \times 4 + 4 \times 8 + 2 \times 5 )/17\) = 59/17

By contrast optimal **B-tree** with \(m=5\) would have an average search time of 30/17 (see below)
- 4 keys @ level 1 accommodate
- 5 nodes @ level 2 can accommodate 20 keys but only need 13 keys (17 - 4)
- Average search time = \((4 \times 1 + 13 \times 2)/17 = 30/17\)

A possible allocation