## Hash Indexes

**a.** Sketch the extensible hash index, and its directory, that results from the following insertions, *where four data entries can fit in a disk page*. Initially the *directory* has two entries and therefore uses just the last bit of the hash value. The initial state of the index, and its global and local depths are shown below. The index entries show the last eight bits of the hash values (not the search key values). Sketch the index and its directory as it appears *after* the insertion of each search key marked with **[sketch]**. For full marks you must show the global depth of the directory, the local depth of each bucket and which bucket is referred to by each directory entry.

```
global depth = 1, both buckets have a local depth of 1
00-----> 0011 1110, 1101 0110, 0001 1010
01-----> 1111 1101, 1011 0011, 0011 1001, 1110 1101
```

Search keys with the following ten hash values are to be inserted:

```
0011 0011 [sketch]
0101 0111
1011 1100
1001 0101
0110 0001 [sketch]
1111 1001
0000 1111
1001 0110 [sketch]
0110 0000
1000 0110 [sketch]
```

| dir. **gd = 2** | bucket | ld |
|---|---|---|
| 00 | 0011 1110, 1101 0110, 0001 1010 | 1 |
| 01 | 1111 1101, 0011 1001, 1110 1101 | **2** |
| **10->00** | | **1** |
| **11** | 1011 0011, **0011 0011** | **2** |

Bucket 1 overflows causing the directory to double and bucket 11 to be created

| dir. **gd = 3** | bucket | ld |
|---|---|---|
| 000 | 0011 1110, 1101 0110, 0001 1010, **1011 1100** | 1 |
| 001 | 0011 1001, **0110 0001** | **3** |
| 010->000 | | 1 |
| 011 | 1011 0011, 0011 0011, **0101 0111** | 2 |
| **100->000** | | **1** |
| **101** | 1111 1101, 1110 1101, **1001 0101** | **3** |
| **110->000** | | **1** |
| **111->011** | | **2** |

Bucket 01 overflows causing the directory to double and bucket 101 to be created

| dir. gd = 3 | bucket | ld |
|---|---|---|
| 000 | 1011 1100 | **2** |
| 001 | 0011 1001, 0110 0001, **1111 1001** | 3 |
| **010** | 0011 1110, 1101 0110, 0001 1010, **1001 0110** | **2** |
| 011 | 1011 0011, 0011 0011, 0101 0111, **0000 1111** | 2 |
| **100->000** | | **2** |
| 101 | 1111 1101, 1110 1101, 1001 0101 | 3 |
| **110->010** | | **2** |
| 111->011 | | 2 |

Bucket 000 overflows causing bucket 100 to be created

| dir. gd = 3 | bucket | ld |
|---|---|---|
| 000 | 1011 1100, **0110 0000** | 2 |
| 001 | 0011 1001, 0110 0001, 1111 1001 | 3 |

Bucket 010 overflows causing bucket 110 to be created

| | | |
|---|---|---|
| 010 | 0001 1010 | **3** |
| 011 | 1011 0011, 0011 0011, 0101 0111, 0000 1111 | 2 |
| 100->000 | | 2 |
| 101 | 1111 1101, 1110 1101, 1001 0101 | 3 |
| **110** | 0011 1110, 1101 0110, 1001 0110, **1000 0110** | **3** |
| 111->011 | | 2 |

**b.** Sketch the linear hash index that results from the seven initial data entries and subsequent ten insertions given in (a) above, where the maximum average occupancy for index buckets is 80%. Sketch the index just after the insertion of the search keys marked with [sketch] (above).

Note that there is no directory; the first column is shown to indicate the index of the primary bucket. The threshold ratio of data entries / buckets if 80% of the 4 (the number of entries per block) = 3.2. Note that at the start 7/2 =3.5 so the starting index has three buckets.

| | primary block | overflow block |
|---|---|---|
| 00 | | |
| 01 | 1111 1101, 0011 1001, 1110 1101, 1011 0011 | **0011 0011** |
| 10 | 0011 1110, 1101 0110, 0001 1010 | |

The initial index has three buckets, the 8th entry goes into an overflow block

| | primary block | overflow block |
|---|---|---|
| 00 | **1011 1100** | |
| 01 | 1111 1101, 0011 1001, 1110 1101, **1001 0101** | **0110 0001** |
| 10 | 0011 1110, 1101 0110, 0001 1010 | |
| **11** | 1011 0011, 0011 0011, **0101 0111** | |

After the 10th entry (1011 1100) r/b = 3.33 so the fourth bucket (11) is created.

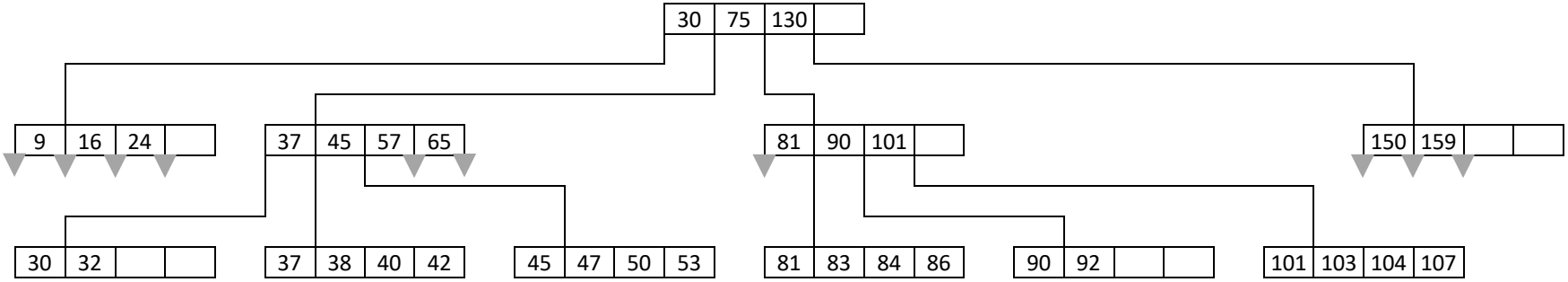
| | primary block | overflow block |
|---|---|---|
| 000 | | |
| 001 | 1111 1101, 0011 1001, 1110 1101, 1001 0101 | 0110 0001, **1111 1001** |
| 010 | 0011 1110, 1101 0110, 0001 1010, **1001 0110** | |
| 011 | 1011 0011, 0011 0011, 0101 0111, **0000 1111** | |
| **100** | 1011 1100 | |

After the 13th entry (1111 1001) r/b = 3.25 so the fifth bucket (100) is created.

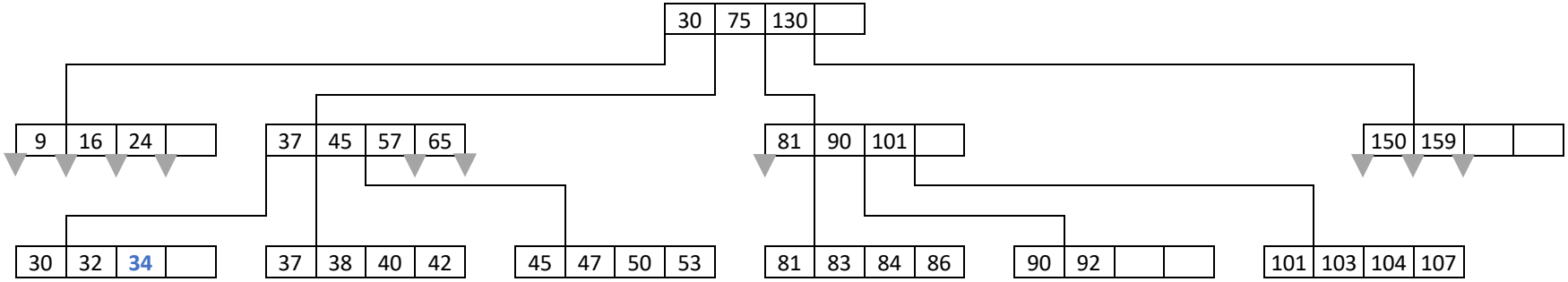
| | primary block | overflow block |
|---|---|---|
| 000 | **0110 0000** | |
| 001 | 0011 1001, 0110 0001, 1111 1001 | |
| 010 | 0011 1110, 1101 0110, 0001 1010, 1001 0110 | **1000 0110** |
| 011 | 1011 0011, 0011 0011, 0101 0111, 0000 1111 | |
| 100 | 1011 1100 | |
| 101 | 1111 1101, 1110 1101, 1001 0101 | |

After the 17th entry (1111 1001) r/b = 3.4 so the sixth bucket (101) is created.

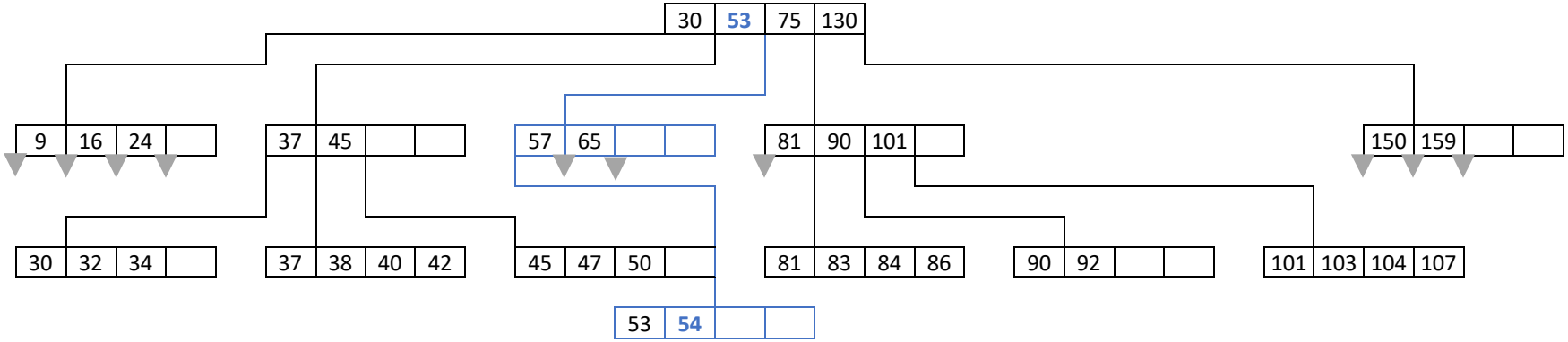
## B+ Tree Insertions



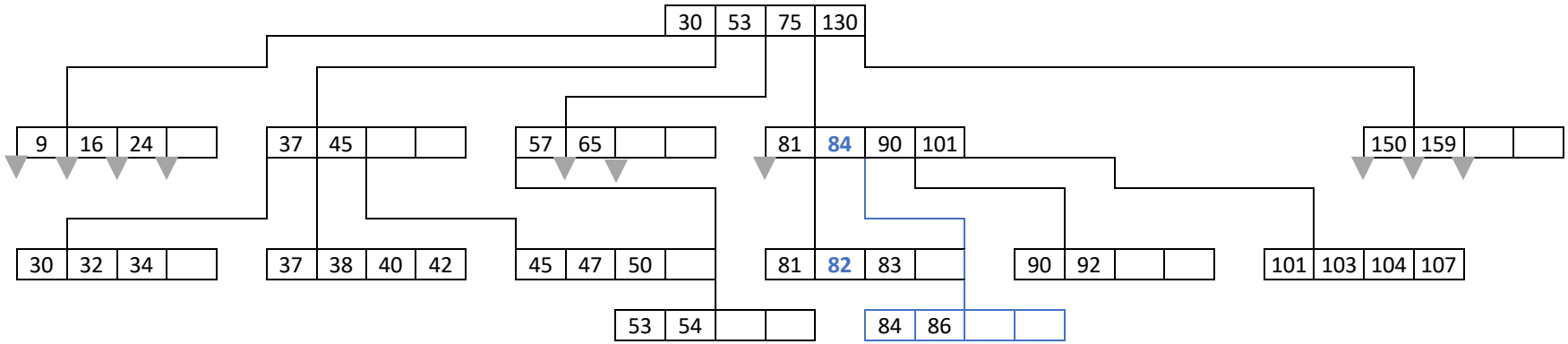

Insert 34



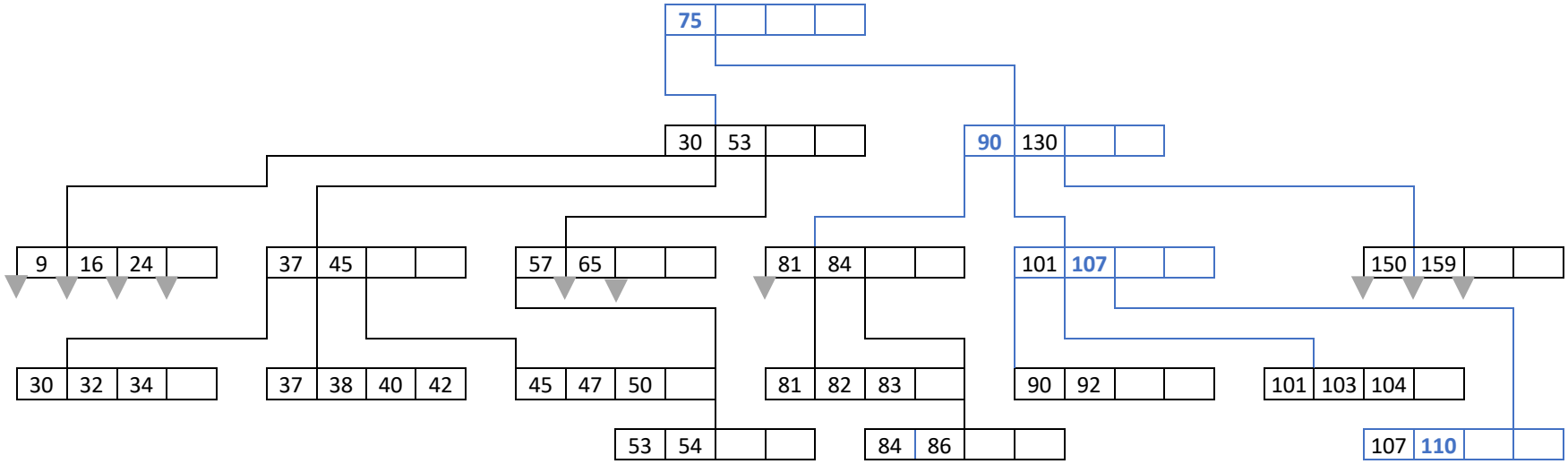

Insert 54



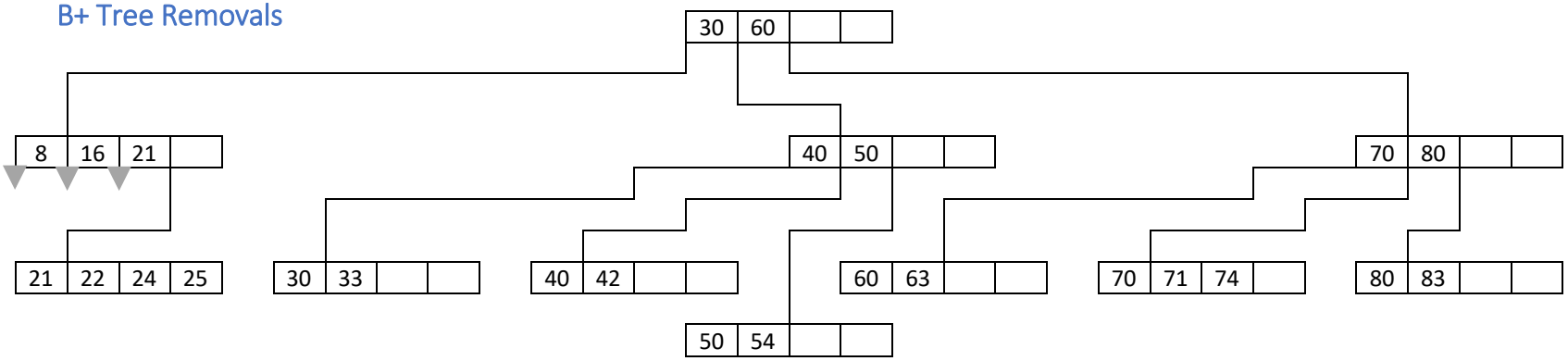
Insert 82



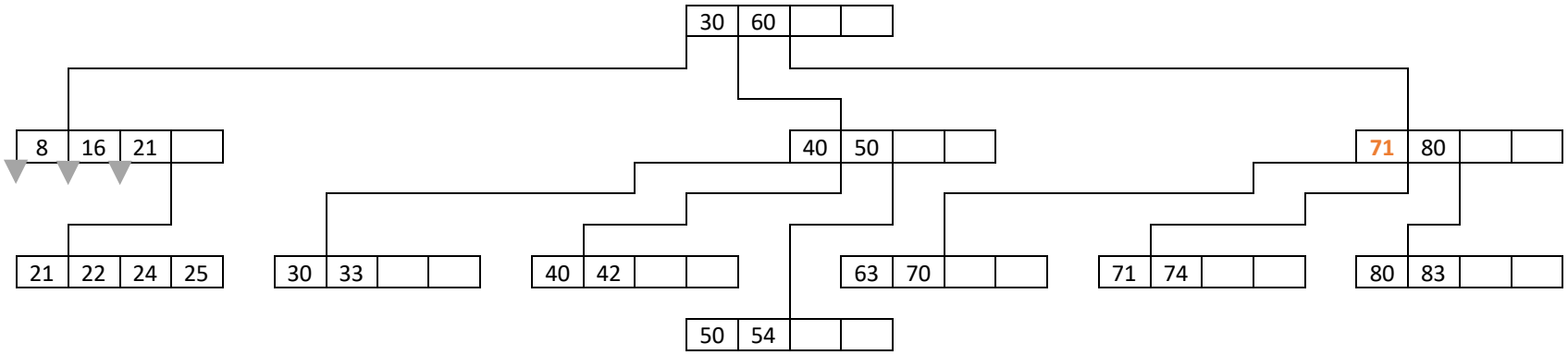

Insert 110



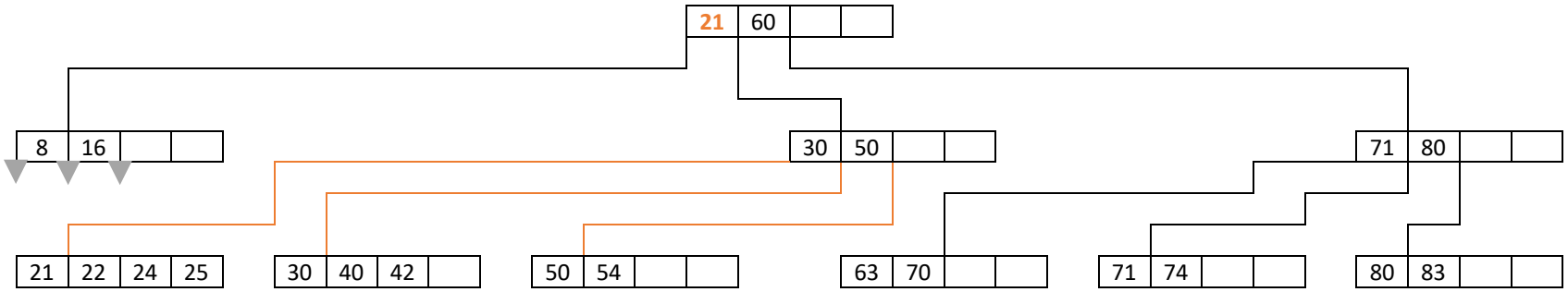
## B+ Tree Removals



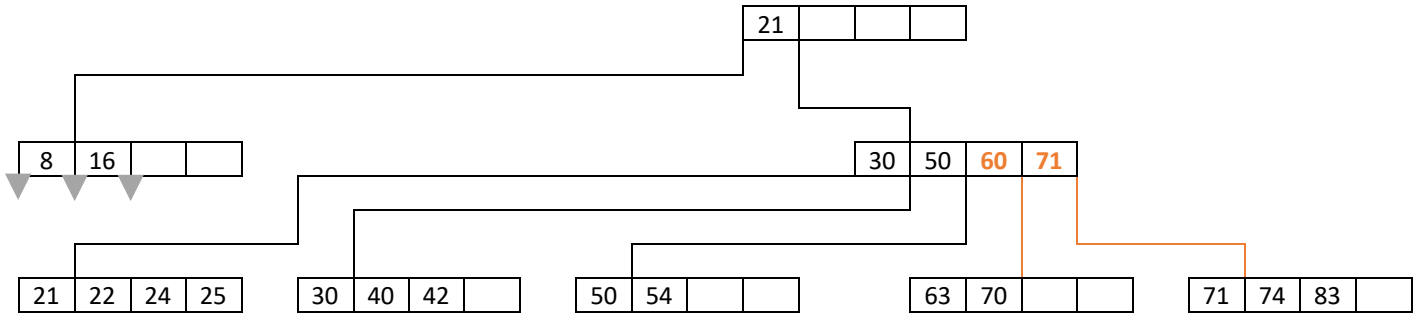

Delete 60



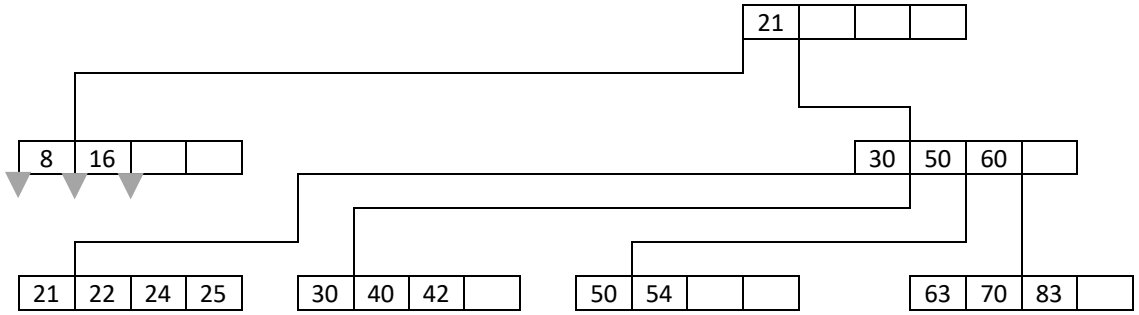

Delete 33



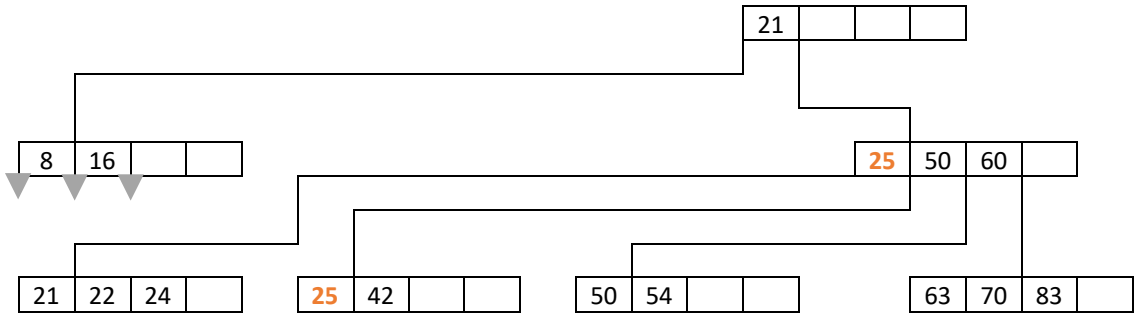
Delete 80



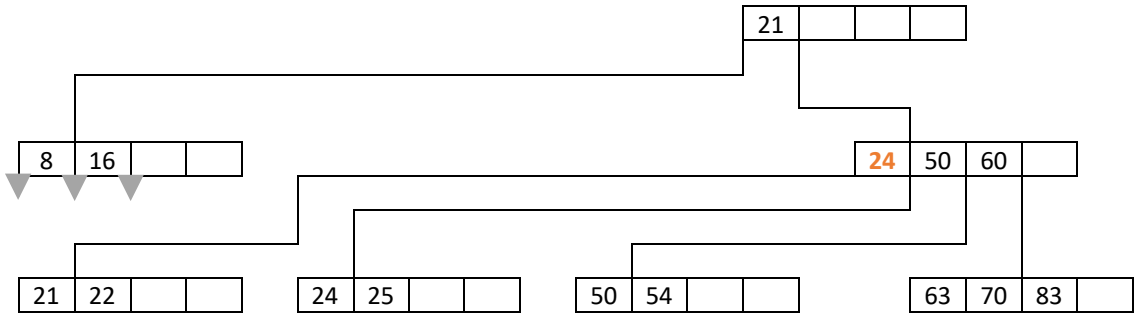

Delete 71 then 74



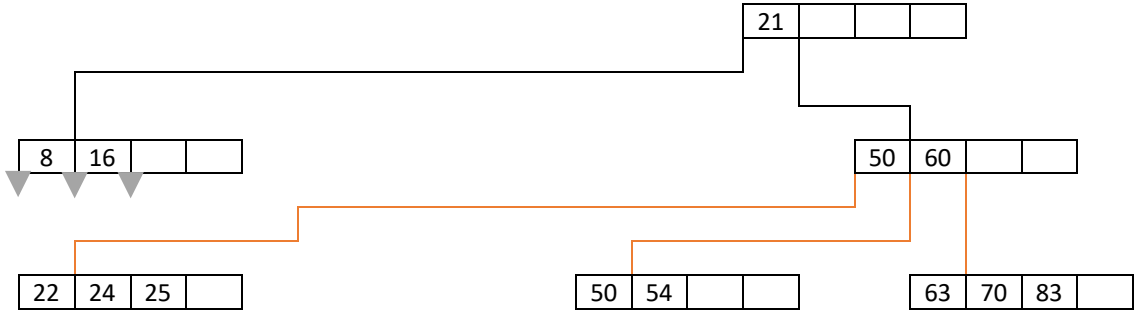

Delete 30 then 40



Delete 42




Delete 21




Delete 70 then 83