**ALGORITHM FOR SKIP LIST**

**Algorithm 1: Create a New Skip List Node**

**Input:** Key key to be inserted, level level of the node.

**Output:** A new Skip List node.

**Steps:**

1.newnode = new SkipListNode(key, level)

2.For i from 0 to level, do:

2.1 newnode->forward[i] = nullptr

3.return newnode

**Algorithm 2: Insert a Key into the Skip List**

**Input**: Key key to be inserted into the Skip List.

**Output:** Node inserted and list structure possibly altered.

**Steps:**

1.If header->forward[0] is nullptr or key < header->forward[0]->key or key > last node's key,

2.temp = header

3.int newLevel = randomLevel()

4.SkipListNode\* update[MAX\_LEVEL+1]

4.1 Initialize all update to nullptr

4.2 For i from level down to 0, do:

4.2.1 While temp->forward[i] != nullptr and temp->forward[i]->key < key:

4.2.1.1 temp = temp->forward[i]

4.3 update[i] = temp

5.If newLevel > level, update the header for new levels.

6.newNode = createNode(key, newLevel)

6.1 For i from 0 to newLevel, do:

6.1.1 newNode->forward[i] = update[i]->forward[i]

6.2 update[i]->forward[i] = newNode

7.Return newNode

**Algorithm 3: Search for a Key in the Skip List**

**Input:** Key to search in the Skip List

**Output:** true if the key exists, false otherwise.

**Steps:**

1.temp = header

2.For i from level down to 0, do:

2.1 While temp->forward[i] != nullptr and temp->forward[i]->key < key:

2.2 temp = temp->forward[i]

3.temp = temp->forward[0]

4. If temp != nullptr and temp->key == key:

4.1 Return true

5. Else:

5.1 Return false

**Algorithm 4: Delete a Key from the Skip List**

**Input**: Key key to be deleted from the Skip List.

**Output:** true if the node was successfully deleted, false if the node was not found.

**Steps:**

1.SkipListNode\* update[MAX\_LEVEL+1]

1.1 Set all update to nullptr

2.For i from level down to 0, do:

2.1 While temp->forward[i] != nullptr and temp->forward[i]->key < key:

2.1.1 temp = temp->forward[i]

2.2 update[i] = temp

3.temp = temp->forward[0]

4. If temp and temp->key == key:

4.1 For i from 0 to level, do:

4.1.1 If update[i]->forward[i] == temp:

4.1.1.1 update[i]->forward[i] = temp->forward[i]

4.1.1.2 Decrease level if necessary

5. Delete temp

5.1 Return true

6. Else:

6.1 Return false