30 Sept, 2020

NIHA
IBM18CS060
S-B

1. Insert function:-

```
void skiplist :: insert ( int key) {
    Node * curr = head;
    Node * update [MaxLvl +1];
    memset ( update, 0, sizeof (Node *)*(MaxLvl+1));
    for ( int i= level ; i >=0 ; i--)
    {
     while ( curr → forward [i] != NULL &&
          curr → forward [i] → key < key)
      {
        curr = curr → forward [i] ;
      }
      update [i] = curr ;
    }
    curr = curr → forward [0];
    if ( curr == NULL || curr → key != key)
    {    int rlvl = randomlevel ();
         if ( rlvl > level)
        {
          for ( int i= level +1; i<rlvl +1 ; i++)
                update[i] = head;
                level = rlvl.
        }
      Node *neon = createNode (key, rlvl);
      for ( int i=0 ; i< = rlvl ; i++) {
        neon → forward [i] = update [i] → forward [i];
        update [i] → forward [i] = n ;
      }
    }
};
```

```
void SkipList :: delete (int key)
{
    Node *curr = head;
    Node *update [MaxLvl +1];
    memset (update, 0, sizeof (Node *) * (MaxLvl +1));
    for ( int i= level ; i>=0; i--)
    {
        while (curr → forward[i] != NULL &&
                curr → forward[i] → key < key)
        {
            curr = curr → forward[i];
        }
        update [i] = curr;
    }
    curr = curr → forward [0];
    if (curr != NULL && curr → key == key)
    {
        for (int i=0; i<=level ; i++)
        {
            if ( update [i] → forward [i] != curr)
                break;
            update [i] → forward [i] = curr → forward[i];
        }
    while ( level >0 && head → forward[level] == 0)
        level --;
    }
};
```

```
void skiplist::search(int key){
    Node *curr = head;
    for(int i= level; i>=0; i--)
    {
        while(curr->forward[i] != NULL &&
              curr->forward[i]->key<key)
            curr = curr->forward[i];
    }
    curr = curr->forward[0];
    if(curr!=NULL && curr->key==key)
        cout<<"Found:"<<key<<endl;
};
```