Page No.
Date: / /

M.S. SATHWICK KIRAN
1BM18CS050

```
int randomlevel () {
float r = (float) rad () / RAND_MAX;
 int lvl = 0;
while (r<P && Lvl < MAX LVL)
 { lvl ++;
    r = (float) rad () /RAND_max;
 } return lvl;
 };
     void insertElemen (int key)
 { Node * current : header;
   Node * update (MAX.LVL++);
   memset (update, 0, size of (Node *) * (MAXLVL ++));
   for (int i = level; i>=0; i--) {
     while (current -> forward (i) ! = NULL && Current -> forward (i) ->
                                                  Key < key)
    Current ~~:~~ : Current -> forward (i);
     update (i) = Current; }
         current = current -> forward [0];
       if (current == NULL || current -> Key ! = Key) {
        int rlevel = random Level ();
        if (rlevel > level) {
        for (int i = level+1; i< rlevel +1; i++)
          update [i] = header;
          level = rlevel;  }
     Node * n = ~~Ex~~ Create Node (Key, rlevel);
      for (int i = 0; i <= rlevel; i++) {
        n -> forward [i] = update [i] -> forward [i];
     update [i] -> forward [i] = n; }
        Cout << " Successfully Inserted key " << key << \n";
   }};
```

Mayur

```
void deleteElement (int key) {
Node * Current = header;
Node * update (MAX LVL ++);
memset (update, 0 size of (Node *) * (MAX LVL ++));
 for (int i = level; i>=0; i--) {
  while (Current → forward [i] ! = NULL && current → forward[i] →
                                                        key < key)
 current = current → forward [i];
   update (i) = current; }
    current = Curret → forward (0);
   if (Curret != NULL and current → key == key)
    {
      for (int i = 0; i<=level ; i++) {
       if (update (i) → forward (i) != current)
          break;
        update (i) → forward (i) = current → forward [i];
     }
      while (level > 0 && header → forward (level) == 0)
       level --;
      cout << "Deleted "<< key << "\n" }};
void search Ele (int key)
     {
         Node * curret = header;
         for (int i = level ; i>=0; i--){
      while (current → forward [i] && Curret → forward [i] →
                                                      key < key)
   Curret = current → forward (i) ;}
    Curret = curret → forward (o);
       if Curret and Curret → key ==key)
         Cout << "Found :"<< key << "\n";
   };
```