Prathik Kavanth
1BM18CS139

```
int randomlevel()
{
  float r = (float) rand()/rand_max;
  int lvl = 0;
  while(r<P && lvl < MaxLvl)
  {
     lvl++;
     r = (float) rand()/rand_max;
  }
  return lvl;
};

void insert(int key)
{
  Node *current = header;
  Node *update[MAXLVL+1];
  memset(update, 0, sizeof(Node*)*(MAXLVL+1));
  for(int i = level; i>=0; i--)
  { while(curr->forw[i] != NULL &&
          curr->forw[i]->key < key)
      current = current->forward
      curr = curr->forw[i];
    update[i] = curr;
  }
   curr = curr->forward[0];
  if(current == NULL || curr->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 = createnode (key, rlevel);
  for (int i=0; i<=rlevel; i++)
  {  n->forw[i] = update[i] -> forw[i];
     update[i] -> forw[i] = n;
  }
  cout << "Inserted" << key << "\n";
  }
};
void delete (int key)
{ Node *curr = header;
  Node *update[MAXLVL+1];
  memset (update, 0, sizeof (Node*) *(MAXLVL+1));
for (int i = level; i>=0; i--)
{
  while (curr -> forw[i] != NULL && curr->forw[i]
        ->key < key)
      curr = curr -> forw [i];
      update [i] = curr;
}
curr = curr -> forw [0];
if (curr != NULL and curr->key == key)
{ for (int i=0; i<=level; i++)
  { if (update [i]-> forw [i] != curr)
      break;
    update[i] -> forw[i] = curr -> forw[i];
}
```

```
        while (level > 0 && header -> forw [level] == 0)
            level --;
        cout << "deleted " << key << "\n";
    }
};
void search (int key)
{
    Node * curr = header;
    for (int i = level; i >= 0; i--)
    {   while (curr -> forw [i] &&
                curr = curr -> forw [i];
    }
    curr = curr -> forw [0];
    if (curr and curr -> key == key)
        cout << "found" << key;
};
```