```
#define KEY_INVAL 0
#define KEY_VALID 1
#define KEY_INSER 2

#define KEY_BUCKT 4

struct bucket
{
  union
  {              // snapshot and map point to the same memory location
    uint64_t snapshot;     //address all 8 bytes of the map with 1 CAS
    struct
    {
      uint32_t version;
      uint8_t  map[KEY_BUCKT];
    };
  };
  uint64_t key[KEY_BUCKT];
  void*    val[KEY_BUCKT];
};

/* help functions ************************************************************************* */
uint64_t setmap(snapshot, index, val); /* returns a new snapshot, where map[index] set to val*/
int getempty(snapshot); /* returns index i, where map[i] == KEY_INVAL */
uint64_t setmap_and_incversion(snapshot, index, val); /* setmap & version++ */


/* ht operations ************************************************************************** */

/* returns the found element or NULL */
void* search(bucket* b, uint64_t key)
{
  for (int i = 0; i < KEY_BUCKT; i++)
    {
      void* val = b->val[i];
      if (b->map[i] == KEY_VALID && b->key[i] == key) /* disregard any invalid keys */
        {
          if (b->val[i] == val)
            return val;
          else
            return NULL;
        }
    }
}


/* returns the removed element or NULL */
void* remove(bucket* b, uint64_t key)
{
 retry:
  snapshot_t s = b->snapshot;
  for (int i = 0; i < KEY_BUCKT; i++)
    {
      if (b->key[i] == key && s.map[i] == KEY_VALID) /* use the map of snapshot s */
        {
          void* removed = b->val[i];
          uint64_t s1 = setmap(s, i, KEY_INVAL);
          if (CAS(&b->snapshot, s, s1))
            return removed;
          else
            goto retry;
        }
    }

  return NULL;
}
```

```
/* returns true if inserted, else false */
bool put(bucket* b, uint64_t key, void* val)
{
  int empty = -2;
 retry:
  uint64_t s = b->snapshot;

  int i;
  for (i = 0; i < KEY_BUCKT; i++)
    {
      void* val = b->val[i];
      if (b->map[i] == KEY_VALID && b->key[i] == key) /* disregard any invalid keys */
        {
          if (b->val[i] == val)
            {
              if (empty >= 0) /* release the map position of a previous try */
                {
                  b->map[empty] = KEY_INVAL;
                }
              return false;     /* key already in the bucket */
            }
        }
    }

  if (empty < 0)
    {
      empty = getempty(s);
      uint64_t s1 = setmap(s, empty, KEY_INSER);
      if (CAS(b->snapshot, s, s1) == false)
        {
          empty = -2;           /* indicated that we don’t hold a map position */
          goto retry;
        }

      b->val[empty] = val;
      b->key[empty] = key;
    }
  else                          /* just update the s1 based on the new s */
    uint64_t s1 = setmap(s, empty, KEY_INSER);

  uint64_t s2 = setmap_and_incversion(s, empty, KEY_VALID);
  if (CAS(b->snapshot, s1, s2) == false)
      goto retry;

  return true;
}
```