Solution 3: The C `multitable`

a)

```
typedef struct {
   hashset mappings;
   int keySize;
   int valueSize;
} multitable;

void MultiTableNew(multitable *mt, int keySizeInBytes, int valueSizeInBytes,
                   int numBuckets, MultiTableHashFunction hash,
                   MultiTableCompareFunction compare)
{
   mt->keySize = keySizeInBytes;
   mt->valueSize = valueSizeInBytes;
   HashSetNew(&mt->mappings, keySizeInBytes + sizeof(vector), numBuckets,
              hash, compare, NULL);
}
```

b)

```
void MultiTableEnter(multitable *mt, const void *keyAddr, const void *valueAddr)
{
   char buffer[mt->keySize + sizeof(vector)];
   vector *values;
   void *found = HashSetLookup(&mt->mappings, keyAddr);
   if (found == NULL) {
      memcpy(buffer, keyAddr, mt->keySize);
      values = (vector *)(buffer + mt->keySize);
      VectorNew(values, mt->valueSize, NULL, 0);
      VectorAppend(values, valueAddr);
      HashSetEnter(&mt->mappings, buffer);
   } else {
      values = (vector *)((char *) found + mt->keySize);
      VectorAppend(values, valueAddr);
   }
}
```

c)

```
typedef struct {
   MultiTableMapFunction map;
   void *auxData;
   int keySize;
} maphelper;
```

```
static void HashSetMapper(void *elem, void *auxData)
{
   int i;
   maphelper *helper = auxData;
   vector *values = (vector *)((char *) elem + helper->keySize);
   for (i = 0; i < VectorLength(values); i++)
      helper->map(elem, VectorNth(values, i), helper->auxData);
}

void MultiTableMap(multitable *mt, MultiTableMapFunction map,
                   void *auxData)
{
   maphelper helper = {map, auxData, mt->keySize};
   HashSetMap(&mt->mappings, HashSetMapper, &helper);
}
```

Solution 4: `multitable` Client Code

```
void ListRecordsInRange(multitable *zipCodes, char *low, char *high)
{
   char *endpoints[] = {low, high};
   MultiTableMap(zipCodes, InRangePrint, endpoints);
}

static void InRangePrint(void *keyAddr, void *valueAddr, void *auxData)
{
   char *zipcode = (char *) keyAddr;
   char *city = *(char **) valueAddr;
   char **endpoints = (char **) auxData;
   char *low = endpoints[0];
   char *high = endpoints[1];

   if ((strcmp(zipcode, low) >= 0) && (strcmp(zipcode, high) <= 0))
      printf("%5s: %s\n", zipcode, city);
}
```