```
// tuple.c ... functions on tuples
// part of Multi-attribute Linear-hashed Files
// Last modified by John Shepherd, July 2019

#include "defs.h"
#include "tuple.h"
#include "hash.h"
#include "chvec.h"
#include "bits.h"

char *copyString(char *);

// return number of bytes/chars in a tuple

int tupLength(Tuple t)
{
    return strlen(t);
}

// extract values into an array of strings

void tupleVals(Tuple t, char **vals)
{
    char *c = t, *c0 = t;
    int i = 0;
    for (;;) {
        while (*c != ',' && *c != '\0') c++;
        if (*c == '\0') {
            // end of tuple; add last field to vals
            vals[i++] = copyString(c0);
            break;
        }
        else {
            // end of next field; add to vals
            *c = '\0';
            vals[i++] = copyString(c0);
            *c = ',';
            c++; c0 = c;
        }
    }
}

// release memory used for separate attirubte values

void freeVals(char **vals, int nattrs)
{
    int i;
    for (i = 0; i < nattrs; i++) free(vals[i]);
    free(vals);
}
```

```
// hash a tuple using the choice vector

Bits tupleHash(ChVec cv, Count nvals, Tuple t)
{
    Bits mah = 0;
    char **vals = malloc(nvals*sizeof(char *));
    assert(vals != NULL);
    tupleVals(t, vals);

    // compute MAH hash value
    // TODO ... approx 15 lines of code
    Bits hashes[nvals];
    for (int i = 0; i < nvals; i++) {
        hashes[i] = hash_any((unsigned char *)vals[i],
                             strlen(vals[i]));
    }

    for (int i = 0; i < MAXCHVEC; i++) {
        if (bitIsSet(hashes[cv[i].att], cv[i].bit)) {
            mah = setBit(mah, i);
        }
    }

    freeVals(vals, nvals);
    return mah;
}

// puts printable version of tuple in user-supplied buffer

void tupleString(Tuple t, char *buf)
{
    strcpy(buf,t);
}

// make a copy of a string

char *copyString(char *str)
{
    char *buf = malloc(strlen(str)+1);
    assert(buf != NULL);
    strcpy(buf, str);
    return(buf);
}
```