COMP9315 Final Exam

The University of New South Wales
COMP9315 DBMS Implementation
Final Exam

DBMS Implementation

[Instructions] [Notes] [PostgreSQL] [C]
[Q1] [Q2] [Q3] [Q4] [Q5] [Q6] [Q7] [Q8]

Question 1 (16 marks)

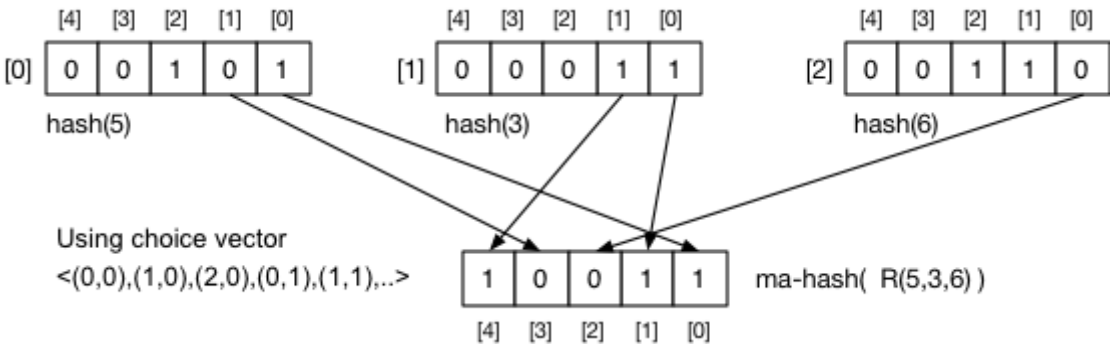
In this question, you need to complete a program that generates a multi-attribute hash (MAH) value, given a choice vector and a tuple.

The code for this question is in the `q1` directory, which contains:

- `mah.c` ... main program for making the hash value
- `bits.c` ... implementation of a bit-string ADT
- `bits.h` ... interface definitions for a bit-string ADT
- `hash.c` ... implementation of a hash function
- `hash.h` ... interface definitions for hash function
- `chvec.c` ... implementation of a choice vector ADT
- `chvec.h` ... interface definitions for choice vector ADT
- `tuple.c` ... implementation of a simple tuple ADT
- `tuple.h` ... interface definitions for simple tuple ADT
- `Makefile` ...for building the program
- `tests/` ... directory containing test cases

As you should remember from Assignment 2, MAH uses bits from the hash values for several attributes to determine an overall hash for a tuple; which bits of which attributes are used in the overall hash is determined by a choice vector

Choice Vector Example: Consider a table `R(a,b,c)` with choice vector <(0,0),(1,0),(2,0),(0,1),(1,1),..>. This tells us that bit 0 (least significant) of the overall hash comes from bit 0 of the first attribute (i.e. attribute "a"); bit 1 of the overall hash comes from bit 0 of the second attribute ("b"), and so on. If we insert a tuple like `(5,3,6)` into the table, and if `hash(5)=00101`, `hash(3)=00011` and `hash(6)=00110`, then the overall hash (assuming 5-bit hash values) will be `10011`, and so the tuple would be inserted into bucket 19. The following diagram shows this example:



All of the code files are complete and fully-functional except for `tuple.c`, which you are required to complete. You should not modify any of the other files in developing your solution, unless you want to add debugging code. Note that you only submit `tuple.c`, so any changes to other files will not be available in the testing environment and will not be marked.

The `mah` command takes two command-line parameters:

- `ChoiceVector` ... a choice vector in the same format as Assignment 2
- `Tuple` ... a tuple as a sequence of comma-separated strings

You can build the `mah` executable by running the `make` command.

Some examples of use:

```
$ ./mah
Usage: ./mah ChoiceVector Tuple
          choice vector                tuple
$ ./mah  "0,0:1,0:2,0:0,1:1,1:2,1"  "a,b,c"
hash(a,b,c) = 01011011 10111100 01111100 10110101
$ ./mah  "0,0:0,1:0,2:1,0:1,1:1,2"  "one,two"
hash(one,two) = 01001110 10011000 01000001 11100001
```

The hash value is based on the hash values of the individual attributes and the choice vector:

```
# Hash values
hash(a) = 00101111 11100001 00011010 01110001
hash(b) = 11101011 01100101 01010001 10111010
hash(c) = 01101100 01000111 11011011 00100111
# Choice vector
0,0:1,0:2,0:0,1:1,1:2,1:0,31:1,31:2,31:0,30:1,30:2,30:
0,29:1,29:2,29:0,28:1,28:2,28:0,27:1,27:2,27:0,26:1,26:
2,26:0,25:1,25:2,25:0,24:1,24:2,24:0,23:1,23
```

Note that entire 32-bit choice vector is formed by taking what is specified on the command-line and then filling in the remaining bits automatically by interleaving the higher-order bits from the attributes.

If you compile and test the code without changes, the hash value will always be all zero.

Your Task: complete the `tupleHash()` function in `tuple.c`

The multi-attribute hash value is computed via the `tupleHash()` function. The function is contained in the `tuple.c` file and is currently a stub that always returns 0. You need to add code in place of the `TODO` comment to compute the multi-attribute hash using the hashes for the individual attributes and the choice vector. Compute the hash for each attribute using the `hash_any()` function.

For debugging, consider displaying the hash value for each individual attribute, displaying the choice vector, and checking that the individual bits have correctly been transferred into the MAH hash value.

The `tests` directory contains a number of test cases for the `mah` program. You can execute an individual test case by running a command like

```
$ sh tests/01.sh
```

which runs the example above.

You should also be able to devise your own test cases easily enough.

To help you check whether your program is working correctly, there is a script called `run_tests.sh` which will run the program against all of the tests and report the results. It will also add the output from your program into the `tests` directory; comparing your output against the expected output might help you to debug your code. You can run the testing script as:

```
$ sh run_tests.sh
```

Once your function is working (passes all tests), follow the submission instructions below. Even if it fails some (or even all) tests, you should submit because you can get *some* marks. If your program does not compile, or if you simply submit the supplied code, then your "answer" is worth zero marks.

Submission Instructions:

- Type your answer to this question into the file called `tuple.c`
- Submit via: **`give cs9315 exam_q1 tuple.c`**
  or via: Webcms3 > exams > Final Exam > Submit Q1 > Make Submission

End of Question