Programming Project #2
EGRE246 Fall 2021
Tuples
(revised 9-23-21)

1 Overview

Selective quoting from the Wikipedia entry for tuple:

> In mathematics, a **tuple** is a finite ordered list (sequence) of elements. An n-tuple is a sequence (or ordered list) of n elements, where n is a non-negative integer. There is only one 0-tuple, referred to as the empty tuple. Mathematicians usually write tuples by listing the elements within parentheses "( )" and separated by commas.

In this project we will create tuples of strings. The number of items in a tuple is referred to as its **arity**. Examples of tuples:

Arity	Examples
0	()
1	("cheese")
2	("hi","mom")
3	("1","2","3")
4	("how","now","brown","cow")
5	("VCU Engineering","","green","Alaska","silly boy")
⋮	

Write a separately compiled C module to implement tuples as specified below. File `ntuples.h` (downloadable off the class web pages; you may not modify this file in any way!):

```
#include <stdbool.h>

#ifndef TUPLE_H
#define TUPLE_H

#define MAX_ARITY 50

struct ttype {
  int arity;
  char *items[MAX_ARITY];
};

typedef struct ttype *ntuple;

/* note that passing NULL for a tuple to any routine is undefined */

char *get(int n,ntuple t);
/* returns the nth item in the tuple t; like arrays in C the
   first item is indexed by 0. Function is undefined if item
```

```
   does not exist */
void put(int n, char *s, ntuple t);
/* inserts s at position n in t, replacing the item (do
   not free it) at n if it already exists. If n == arity,
   tuple increases in size. Adding to a "full" tuple or if n is
   illegal the operation is undefined. Index 0 is the first
   item in the tuple. */

ntuple newTuple(int n,...);
/* returns a new n-tuple; see notes in handout on writing functions
   with varying number of arguments. Examples:
   t = newTuple(0);                    t == ()
   t = newTuple(2,"hi","mom");         t == ("hi","mom")
   t = newTuple(6,"","1","2","3","4","5"); t == ("","1","2","3","4","5") */

void freeTuple(ntuple);
// frees all strings in tuple AND the structure itself

bool equal(ntuple t1, ntuple t2);
/* returns true it t1==t2, false otherwise. Two tuples are
   equivalent if their arities are the same and all items
   in one are identical to the correspoinding items in the
   other */

int arity(ntuple); // returns the arity of the tuple
bool isNtuple(ntuple t,int n); // returns true if tuple has an arity of n

char *ntupToString(ntuple t);
/* returns a string representation of t. Entire tuple should be
   delimited by (), separated by commas, with every item in
   double quotes and no spaces e.g. (), ("go","away","now","").
   String returned should be exactly the correct length to hold
   the tuple. */

#endif
```

2 Variadic Functions in C

A **variadic function** is a function that can accept varying number of arguments. We have used variadic library routines in C, e.g. `printf`, `scanf`.

Example program showing how to write a variadic function in C, downloadable off of the class Canvas pages:

```
#include <stdio.h>
#include <stdarg.h>
#include <math.h>

double average(int n,...) {
   va_list vlist;
   if(n<=0) return NAN;
```

```
   double sum = 0.0;
   va_start(vlist, n);
   for (int i = 0; i < n; i++)
     sum += va_arg(vlist, int);
   va_end(vlist);
   return sum/n;
}

int main(void) {
  printf("Average of 1,2,3,4 = %f\n", average(4, 1,2,3,4));
  printf("Average of 10,20,30,40,50,60,70,80 = %f\n", average(8, 10,20,30,40,50,60,70,80));
  printf("Calling average with no 0 args =  %f\n",average(0));
}
```

3 Deliverables

You should only turn in your tuple module implementation file (i.e. the file containing the implementation of everything defined in `tuple.h`)! This means that the code you submit will not contain a `main` function. I will test your code with my own driver test program. Submit your project (as a single file) via Gradescope.

Due date: Wednesday, September 29