Course: Programming Fundamentals – **ENCM 339**
Lab #: Lab 5
Instructor: S. Norman
Student Name: **Mitchell Sawatzky**
Lab Section: **B02**
Date Submitted: **Oct 13, 2015**

Exercise D

threeVec.c

```
// threeVec.c
// ENCM 339 Fall 2015 Lab 5 Exercise D

#include "threeVec.h"

threeVec_t add_3v(threeVec_t u, threeVec_t v)
{
  threeVec_t result = { u.x + v.x, u.y + v.y, u.z + v.z };
  return result;
}

void scalar_mult(double k, const threeVec_t *src, threeVec_t *dest)
{
    dest->x = k * src->x;
    dest->y = k * src->y;
    dest->z = k * src->z;
}

double dot_product(const threeVec_t *pu, const threeVec_t *pv)
{
    return (pu->x * pv->x) + (pu->y * pv->y) + (pu->z * pv->z);
}

threeVec_t cross_product(const threeVec_t *pu, const threeVec_t *pv)
{
    threeVec_t result = {
        (pu->y * pv->z) - (pu->z * pv->y),
        (pu->z * pv->x) - (pu->x * pv->z),
        (pu->x * pv->y) - (pu->y * pv->x)
    };
    return result;
}

void array_sum(const threeVec_t *a, int n, threeVec_t *result)
{
    int i;

    result->x = 0.0;
    result->y = 0.0;
    result->z = 0.0;

    for (i = 0; i < n; i++) {
        result->x = result->x + a[i].x;
        result->y = result->y + a[i].y;
        result->z = result->z + a[i].z;
    }
}
```

main5D.c

```
// main5D.c
// ENCM 339 Fall 2015 Lab 5 Exercise D

// A small program to do quick checks of the functions
// in the threeVec module.

#include <stdio.h>
#include "threeVec.h"

void print_3v(const threeVec_t *pv, int with_newline)
{
  printf("[%g, %g, %g]", pv->x, pv->y, pv->z);
  if (with_newline)
    fputc('\n', stdout);
```

```
}

int main(void)
{
  threeVec_t a = { 1.0, 2.0, 4.0 }, b = { 1.25, -0.5, 0.375 };
  threeVec_t c;

  // A quick check of add_3v ...
  c = add_3v(a, b);
  printf("Sum of ");
  print_3v(&a, 0);
  printf(" and ");
  print_3v(&b, 0);
  printf(" is  ");
  print_3v(&c, 1);


  // A quick check of scalar_mult ...
  // (Call function to multiply -0.5 times a, putting the result in c,
  // and print a message showing the result.)
  scalar_mult(-0.5, &a, &c);
  printf("Scalar mult of %lf and ", -0.5);
  print_3v(&a, 0);
  printf(" is ");
  print_3v(&c, 1);


  // A quick check of dot_product ...
  // (Call function to find dot product of a and b, putting the result
  // in a variable of type double, and print a message showing the
  // result.)
  double d = dot_product(&a, &b);
  printf("Dot product of ");
  print_3v(&a, 0);
  printf(" and ");
  print_3v(&b, 0);
  printf(" is  ");
  printf("%lf\n", d);

  // A quick check of cross_product ...
  // (Call function to find cross product of a and b, putting the
  // result in c, and print a message showing the result.)
  c = cross_product(&a, &b);
  printf("Cross Product of ");
  print_3v(&a, 0);
  printf(" and ");
  print_3v(&b, 0);
  printf(" is  ");
  print_3v(&c, 1);

  // Set up an array of vectors, and display its contents.
  threeVec_t my_arr[ ] = {
     { 1.0, 2.0, 3.0 },
     { 0.1, 0.2, 0.3 },
     { 0.01, 0.02, 0.03 },
     { 0.001, 0.002, 0.003 },
   };
  size_t my_arr_count = sizeof(my_arr) / sizeof(threeVec_t);
  printf("\nAn array of %zu 3-vectors ...\n", my_arr_count);
  size_t i;
  for (i = 0; i < my_arr_count; i++) {
    printf("  ");
    print_3v(&my_arr[i], 1);
  }


  // A quick check of array_sum ...
  // (Call function to find sum of vectors in my_arr, putting the
  // result in c, and print a message showing the result.)
```

```
  array_sum(my_arr, my_arr_count, &c);
  printf("The vector sum of the vectors in my_arr is ");
  print_3v(&c, 1);

  // Second check of array_sum ... make sure it does the correct
  // thing when the parameter n == 0.
  array_sum(my_arr, 0, &c);
  printf("The vector sum of a 0-length array is ");
  print_3v(&c, 1);


  return 0;
}
```

Terminal Output:

```
Mitchell@ttys000 19:35 {0} [lab5]$ ./test.out
Sum of [1, 2, 4] and [1.25, -0.5, 0.375] is  [2.25, 1.5, 4.375]
Scalar mult of -0.500000 and [1, 2, 4] is [-0.5, -1, -2]
Dot product of [1, 2, 4] and [1.25, -0.5, 0.375] is  1.750000
Cross Product of [1, 2, 4] and [1.25, -0.5, 0.375] is  [2.75, 4.625, -3]

An array of 4 3-vectors ...
  [1, 2, 3]
  [0.1, 0.2, 0.3]
  [0.01, 0.02, 0.03]
  [0.001, 0.002, 0.003]
The vector sum of the vectors in my_arr is [1.111, 2.222, 3.333]
The vector sum of a 0-length array is [0, 0, 0]
```