Course: Programming Fundamentals - **ENCM 339**   
Lab #: Lab 5  
Instructor: M. Moussavi  
Student Name: Liam Wrubleski  
Lab Section: B01  
Date submitted: Oct 23, 2016

# Exercise D

## Macro Text

#define LARGEST\_OF\_THREE(x,y,z) ((x)>(y))?(((x)>(z))?(x):(z)):(((y)>(z))?(y):(z))

# Exercise E

## Code Listing

/\* File: lab5ExE.h

\* ENCM 339 Fall 2016 - lab 5 - Exercise E

\*/

#ifndef lab5ExD\_h

#define lab5ExD\_h

/\* a structure that represents a point on a Cartesian coordinates system. \*/

typedef struct point

{

char label[10]; // a label for a point

double x ; // x coordinate for point in a Cartesian coordinate system

double y; // y coordinate for point in a Cartesian coordinate system

double z; // z coordinate for point in a Cartesian coordinate system

} Point;

Point mid\_point(const Point\* p1, const Point\* p2, const char\* label);

/\* REQUIRES:

\* p1 and p2 point to Point objects

\* PROMISES:

\* returns an object of Point that its x and y coordinates are the middle-

\* point of those objects that p1 and p2 are pointing to. The returned

\* object's label will be the copy of argument label.

\*/

void swap(Point\* p1, Point \*p2);

/\* REQUIRES:

\* p1 and p2 point to objects of Point

\* PROMISES:

\* swaps the values of data members in the two objects \*p1 and \*p2.

\*/

void display\_struct\_point(const Point x);

double distance (const Point\* a, const Point\* b);

/\* REQUIRES:

\* a and b point to objects of Point

\* PROMISES:

\* returns the distance between objects that a and b are pointing to.

\*/

#endif /\* lab5ExD\_h \*/

/\* File: lab5ExE.c

\* ENCM 339 Fall 2016 - lab 5 - Exercise E

\*/

#include "lab5ExE.h"

#include <stdio.h>

#include <math.h>

#include <string.h>

int main(void)

{

Point alpha = { "A1", 2.3, 4.5, 56.0 } ;

Point beta = { "B1", 25.9, 30.0, 97.0 } ;

printf ("Display the values in alpha, and beta: ");

display\_struct\_point(alpha);

display\_struct\_point(beta);

Point \*stp = &alpha;

printf ("\n\nDisplay the values in \*stp: ");

display\_struct\_point(\*stp);

Point gamma = mid\_point(stp, &beta, "M1");

printf ("\n\nDisplay the values in gamma after calling mid\_point function.");

printf ("Expected result is: M1 <14.10, 17.25, 76.50>");

printf("\n\nThe actual result of calling mid\_point function is: ");

display\_struct\_point(gamma);

swap (stp, &beta);

printf ("\n\nDisplay the values in \*stp, and beta after calling swap function.");

printf ("Expected to be:\nB1 <25.90, 30.00, 97.00>\nA1 <2.30, 4.50, 56.00>");

printf("\n\nThe actual result of calling swap function is: ");

display\_struct\_point(\*stp);

display\_struct\_point(beta);

printf("\n\nThe distance between alpha and beta is: %.2f. ", distance(&alpha, &beta));

printf ("(Expected to be: 53.74)");

printf("\nThe distance between gamma and beta is: %.2f. ", distance(&gamma, &beta));

printf ("(Expected to be: 26.87)");

return 0;

}

void display\_struct\_point(const Point x)

{

printf("\n%s <%.2lf, %.2lf, %.2lf>", x.label, x.x, x.y, x.z);

}

Point mid\_point(const Point\* p1, const Point\* p2, const char\* label)

{

double mx = (p2->x + p1->x)/2;

double my = (p2->y + p1->y)/2;

double mz = (p2->z + p1->z)/2;

Point middle = {"?", mx, my, mz};

int i=0;

while(i<9 && label[9]){

middle.label[i] = label[i];

i++;

}

middle.label[i] = '\0';

return middle;

}

void swap(Point\* p1, Point \*p2)

{

Point pt;

pt=\*p1;

\*p1=\*p2;

\*p2=pt;

}

double distance(const Point\* p1, const Point\* p2)

{

double dx = p2->x - p1->x;

double dy = p2->y - p1->y;

double dz = p2->z - p1->z;

return sqrt(pow(dx,2)+pow(dy,2)+pow(dz,2));

}

## Sample Dialogues

liam@Inspiron /cygdrive/c/Users/liam/Documents/GitHub/ENCM339Assignments/lab5

$ ./a

Display the values in alpha, and beta:

A1 <2.30, 4.50, 56.00>

B1 <25.90, 30.00, 97.00>

Display the values in \*stp:

A1 <2.30, 4.50, 56.00>

Display the values in gamma after calling mid\_point function.Expected result is: M1 <14.10, 17.25, 76.50>

The actual result of calling mid\_point function is:

M1 <14.10, 17.25, 76.50>

Display the values in \*stp, and beta after calling swap function.Expected to be:

B1 <25.90, 30.00, 97.00>

A1 <2.30, 4.50, 56.00>

The actual result of calling swap function is:

B1 <25.90, 30.00, 97.00>

A1 <2.30, 4.50, 56.00>

The distance between alpha and beta is: 53.74. (Expected to be: 53.74)

The distance between gamma and beta is: 26.87. (Expected to be: 26.87)

# Exercise F

## Code Listing

// lab5ExF.h

// ENCM 339 - Fall 2016, Exercise F

#ifndef lab5ExF\_h

#define lab5ExF\_h

typedef struct point

{

char label[10];

double x ; // x coordinate for point in a Cartesian coordinate system

double y; // y coordinate for point in a Cartesian coordinate system

double z; // z coordinate for point in a Cartesian coordinate system

}Point;

void reverse (Point \*a, int n);

/\* REQUIRES: Elements a[0] ... a[n-2], a[n-1] exists.

\* PROMISES: places the existing Point objects in array a, in reverse order.

\* The new a[0] value is the old a[n-1] value, the new a[1] is the

\* old a[n-2], etc.

\*/

int search(const Point\* struct\_array, const char\* target, int n);

/\* REQUIRES: Elements struct-array[0] ... struct\_array[n-2], struct\_array[n-1]

\* exists. target points to string to be searched for.

\* PROMISES: returns the index of the element in the array that contains an

\* instance of point with a matching label. Otherwise, if there is

\* no point in the array that its label matches the target-label,

\* it should return -1.

\* If there are more than one match in the struct\_array, returns

\* the index of the first occurrence.

\*/

void display\_struct\_point(const Point x, int i);

void populate\_struct\_array(Point\* array, int n);

#endif /\* lab5ExF\_h \*/

// lab5ExF.c

// ENCM 339 - Fall 2016, Exercise F

#include "lab5ExF.h"

#include <stdio.h>

#include <math.h>

#include<string.h>

int main(void)

{

Point struct\_array[10];

int i;

int position;

populate\_struct\_array(struct\_array, 10);

printf("\nArray of Points contains: \n");

for(i=0; i < 10; i++)

display\_struct\_point(struct\_array[i], i);

printf("\nTest the search function");

position = search(struct\_array, "v0", 10);

if(position != -1)

printf("\nFound: struct\_array[%d] contains %s", position,

struct\_array[position].label);

else

printf("\nstruct\_array doesn't have label: %s.", "v0");

position = search(struct\_array, "E1", 10);

if(position != -1)

printf("\nFound: struct\_array[%d] contains %s", position,

struct\_array[position].label);

else

printf("\nstruct\_array doesn't have label: %s.", "E1");

position = search(struct\_array, "C5", 10);

if(position != -1)

printf("\nFound: struct\_array[%d] contains %s", position,

struct\_array[position].label);

else

printf("\nstruct\_array doesn't have label: %s.", "C5");

position = search(struct\_array, "B7", 10);

if(position != -1)

printf("\nFound: struct\_array[%d] contains %s", position,

struct\_array[position].label);

else

printf("\nstruct\_array doesn't have label: %s.", "B7");

position = search(struct\_array, "A9", 10);

if(position != -1)

printf("\nFound: struct\_array[%d] contains %s", position,

struct\_array[position].label);

else

printf("\nstruct\_array doesn't have label: %s.", "A9");

position = search(struct\_array, "E11", 10);

if(position != -1)

printf("\nFound: struct\_array[%d] contains %s", position,

struct\_array[position].label);

else

printf("\nstruct\_array doesn't have label: %s.", "E11");

position = search(struct\_array, "M1", 10);

if(position != -1)

printf("\nFound: struct\_array[%d] contains %s", position,

struct\_array[position].label);

else

printf("\nstruct\_array doesn't have label: %s.", "M1");

printf("\n\nTesting the reverse function:");

reverse(struct\_array, 10);

printf("\nThe reversed array is:");

for(i=0; i < 10; i++)

display\_struct\_point(struct\_array[i], i);

return 0;

}

void display\_struct\_point(const Point x , int i)

{

printf("\nstruct\_array[%d]: %s <%.2lf, %.2lf, %.2lf>\n",

i, x.label, x.x, x.y, x.z);

}

void populate\_struct\_array(Point\* array, int n)

{

int i;

char ch1 = 'A';

char ch2 = '9';

char ch3 = 'z';

for( i = 0; i < 10; i++)

{

/\* generating some random values to fill them elements of the array: \*/

array[i].x = (7 \* (i + 1) % 11) \* 100 - i /2;

array[i].y = (7 \* (i + 1) % 11) \* 120 - i / 3;

array[i].z = (7 \* (i + 1) % 11) \* 150 - i /4;

if(i % 2 == 0)

array[i].label[0] = ch1++;

else

array[i].label[0] = ch3--;

array[i].label[1] = ch2--;

array[i].label[2] = '\0';

}

}

int search(const Point\* struct\_array, const char\* label, int n)

{

int r=-1;

for(int i = 0; i < n; i++){

int j=0;

if(r != -1) break;

while (struct\_array[i].label[j] || label[j]){

if(struct\_array[i].label[j] != label[j]){

r=-1;

break;

}

r=i;

j++;

}

}

return r;

}

void reverse (Point \*a, int n)

{

for( int i = 0; i < n/2; i++ ){

Point t = a[i];

a[i] = a[n-1-i];

a[n-1-i] = t;

}

}

## Sample Dialogues

liam@Inspiron /cygdrive/c/Users/liam/Documents/GitHub/ENCM339Assignments/lab5

$ ./a

Array of Points contains:

struct\_array[0]: A9 <700.00, 840.00, 1050.00>

struct\_array[1]: z8 <300.00, 360.00, 450.00>

struct\_array[2]: B7 <999.00, 1200.00, 1500.00>

struct\_array[3]: y6 <599.00, 719.00, 900.00>

struct\_array[4]: C5 <198.00, 239.00, 299.00>

struct\_array[5]: x4 <898.00, 1079.00, 1349.00>

struct\_array[6]: D3 <497.00, 598.00, 749.00>

struct\_array[7]: w2 <97.00, 118.00, 149.00>

struct\_array[8]: E1 <796.00, 958.00, 1198.00>

struct\_array[9]: v0 <396.00, 477.00, 598.00>

Test the search function

Found: struct\_array[9] contains v0

Found: struct\_array[8] contains E1

Found: struct\_array[4] contains C5

Found: struct\_array[2] contains B7

Found: struct\_array[0] contains A9

struct\_array doesn't have label: E11.

struct\_array doesn't have label: M1.

Testing the reverse function:

The reversed array is:

struct\_array[0]: v0 <396.00, 477.00, 598.00>

struct\_array[1]: E1 <796.00, 958.00, 1198.00>

struct\_array[2]: w2 <97.00, 118.00, 149.00>

struct\_array[3]: D3 <497.00, 598.00, 749.00>

struct\_array[4]: x4 <898.00, 1079.00, 1349.00>

struct\_array[5]: C5 <198.00, 239.00, 299.00>

struct\_array[6]: y6 <599.00, 719.00, 900.00>

struct\_array[7]: B7 <999.00, 1200.00, 1500.00>

struct\_array[8]: z8 <300.00, 360.00, 450.00>

struct\_array[9]: A9 <700.00, 840.00, 1050.00>