Course: ENSF 337 – Fall 2020

Lab #: Lab 5

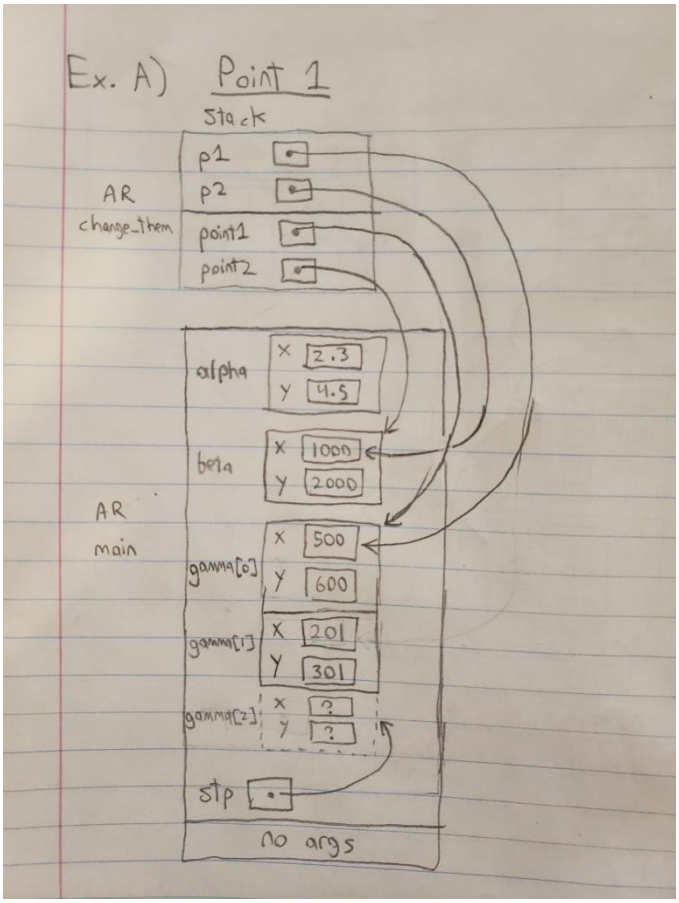
Instructor: M. Moussavi

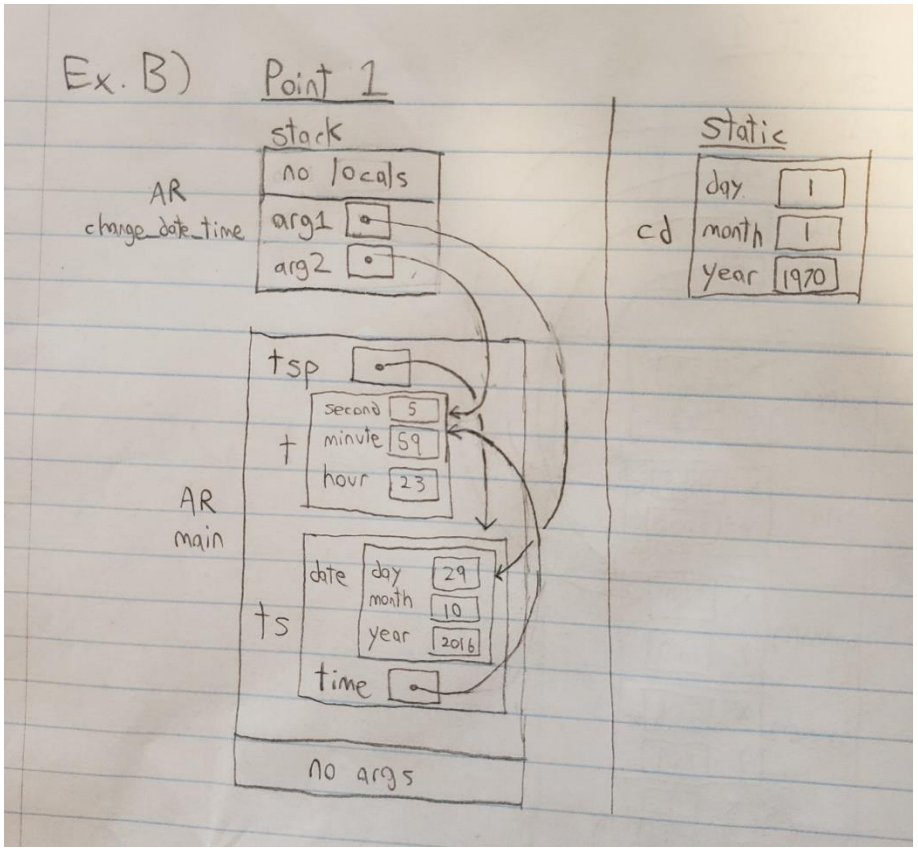
Student Name: Quentin Jennings

Lab Section: B03

Submission Date: 2020-10-18

Exercise A and B:





Exercise D display_multiple_column:

```
void display_multiple_column(const IntVector *intV, int col, const char* output_filename)
{
    int loopTrue = 1; //exit condition for do/while loop
    int indexNum = 0; //index variable used as part of nested for loop - keeps the index from the previous loop, used to print values
    FILE *fp = fopen(output_filename, "w"); //opens the output file for writing and creates file pointer fp

    if(fp == NULL){ //closes program if there's an error opening the file
        printf("Error: cannot open the text file %s. Exiting program.\n", output_filename);
        exit(1);
    }

    if(col > intV -> number_of_data) //if there are more columns than data points, sets the number of columns equal to the number of data points
        col = intV -> number_of_data;

    //nested print loop
    do{ //each loop marks a new row
        for(int j = 0; j < col; j++) //each j index marks a new column in the row
        {
            if(indexNum >= (intV -> number_of_data)){ //breaks loop if the index passes the number of data (index starts at 0 so this statement works)
                loopTrue = 0;
                break;
            }
            fprintf(fp, "%10d\t", intV -> storage[indexNum]); //prints the current index in the storage followed by a tab
            indexNum++; //increments index
        }
        fprintf(fp, "\n"); //starts new line at end of row
    }while(loopTrue); //loops until specific break condition is met

    fclose(fp); //closes file when done
}
```

Exercise D lab5exe_D_output.txt:

234	678	999	234
33	22	99	222
45	56	44	77
92	91	81	73
19	18	17	666
555	1	3	6

Exercise E Output:

```
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 E Source Code: (lab5exE.h)

```
1  /* File: lab5exE.h
2   * ENSF Fall 2020 - lab 5 - Exercise E
3   */
4
5  #ifndef lab5ExE_h
6  #define lab5ExE_h
7
8  /* a structure that represents a point on a Cartesian coordinates system. */
9  typedef struct point
10 {
11     char label[10];  // a label for a point
12     double x ;       // x coordinate for point in a Cartesian coordinate system
13     double y;        // y coordinate for point in a Cartesian coordinate system
14     double z;        // z coordinate, added
15 } Point;
16
17 Point mid_point(const Point* p1, const Point* p2, const char* label);
18 /* REQUIRES:
19  *   p1 and p2 point to Point objects
20  * PROMISES:
21  *   returns an object of Point that its x and y coordinates are the middle-
22  *   point of those objects that p1 and p2 are pointing to. The returned
23  *   object's label will be the copy of argument label.
24  */
25
26 void swap(Point* p1, Point *p2);
27 /* REQUIRES:
28  *   p1 and p2 point to objects of Point
29  * PROMISES:
30  *   swaps the values of data members in the two objects *p1 and *p2.
31  */
32
33 void display_struct_point(const Point x);
34
35 double distance (const Point* a, const Point* b);
36 /* REQUIRES:
37  *   a and b point to objects of Point
38  * PROMISES:
39  *   returns the distance between objects that a and b are pointing to.
40  */
41 #endif /* lab5ExE_h */
42
```

Exercise E Source Code: (lab5exE.c)

```c
/* File: lab5exE.c
 * ENSF Fall 2020 - 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 midX, midY, midZ;
    int i;

    //calculations for middle coordinates: uses if/else statements depending on which of the two coordinates is higher to calculate midpoint accordingly
    if(p1 -> x < p2 -> x)
        midX = p1 -> x + (p2 -> x - p1 -> x) / 2;
    else
        midX = p2 -> x + (p1 -> x - p2 -> x) / 2;

    if(p1 -> y < p2 -> y)
        midY = p1 -> y + (p2 -> y - p1 -> y) / 2;
    else
        midY = p2 -> y + (p1 -> y - p2 -> y) / 2;

    if(p1 -> z < p2 -> z)
        midZ = p1 -> z + (p2 -> z - p1 -> z) / 2;
    else
        midZ = p2 -> z + (p1 -> z - p2 -> z) / 2;

    Point middle;

    for(i = 0; label[i] != '\0'; i++) //simple loop that copies label into middle
        middle.label[i] = label[i];
    middle.label[i] = '\0'; //then adds a null at the end to make it a proper string

    //sets the Point's coordinates to the calculated midpoints
    middle.x = midX;
    middle.y = midY;
    middle.z = midZ;

    return middle;
}

void swap(Point* p1, Point *p2)
{
    int i;
    Point temp;

    //copies p1 into a temporary point
    temp.x = p1 -> x;
    temp.y = p1 -> y;
    temp.z = p1 -> z;

    for(i = 0; p1 -> label[i] != '\0'; i++)
        temp.label[i] = p1 -> label[i];
    temp.label[i] = '\0';

    //copies p2 into p1
    p1 -> x = p2 -> x;
    p1 -> y = p2 -> y;
    p1 -> z = p2 -> z;

    for(i = 0; p2 -> label[i] != '\0'; i++)
        p1 -> label[i] = p2 -> label[i];
    p1 -> label[i] = '\0';

    //copies temp into p2
    p2 -> x = temp.x;
    p2 -> y = temp.y;
    p2 -> z = temp.z;

    for(i = 0; temp.label[i] != '\0'; i++)
        p2 -> label[i] = temp.label[i];
    p2 -> label[i] = '\0';
}

double distance(const Point* p1, const Point* p2)
{
    //a simple equation using math functions and pointer notation can replicate the calculation easily in one line, nice!
    return sqrt(pow(((*p1).x - (*p2).x), 2) + pow(((*p1).y - (*p2).y), 2) + pow(((*p1).z - (*p2).z), 2));
}
```

Exercise F Output:

```
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>
```

Exercise F Source Code: (rest of code is same, only asked to make the 2 functions)

```
116  int search(const Point* struct_array, const char* label, int n)
117  {
118
119      //loops through each Point index
120      for(int i = 0; i < n; i++)
121      {
122          for(int j = 0;; j++) //loops through each character in both labels
123          {
124              //if the letters of the label aren't equal to each other, break
125              if(struct_array[i].label[j] != label[j])
126                  break;
127              //if the ends of both labels have been reached, return the index of the Point
128              else if(( struct_array[i].label[j] == '\0') && (label[j] == '\0'))
129                  return i;
130          }
131      }
132
133      return -1; //if nothing was found in the for loop, returns -1
134  }
135
136  void reverse (Point *a, int n)
137  {
138      Point temp[n]; //temporary array of points with n entries
139
140      for(int i = 0; i < n; i++) //sets temp's entries equal to the reverse order of a's entries
141          temp[i] = a[9 - i];
142
143      for(int i = 0; i < n; i++) //sets a's entries equal to the reversed temp array's entries
144          a[i] = temp[i];
145  }
```