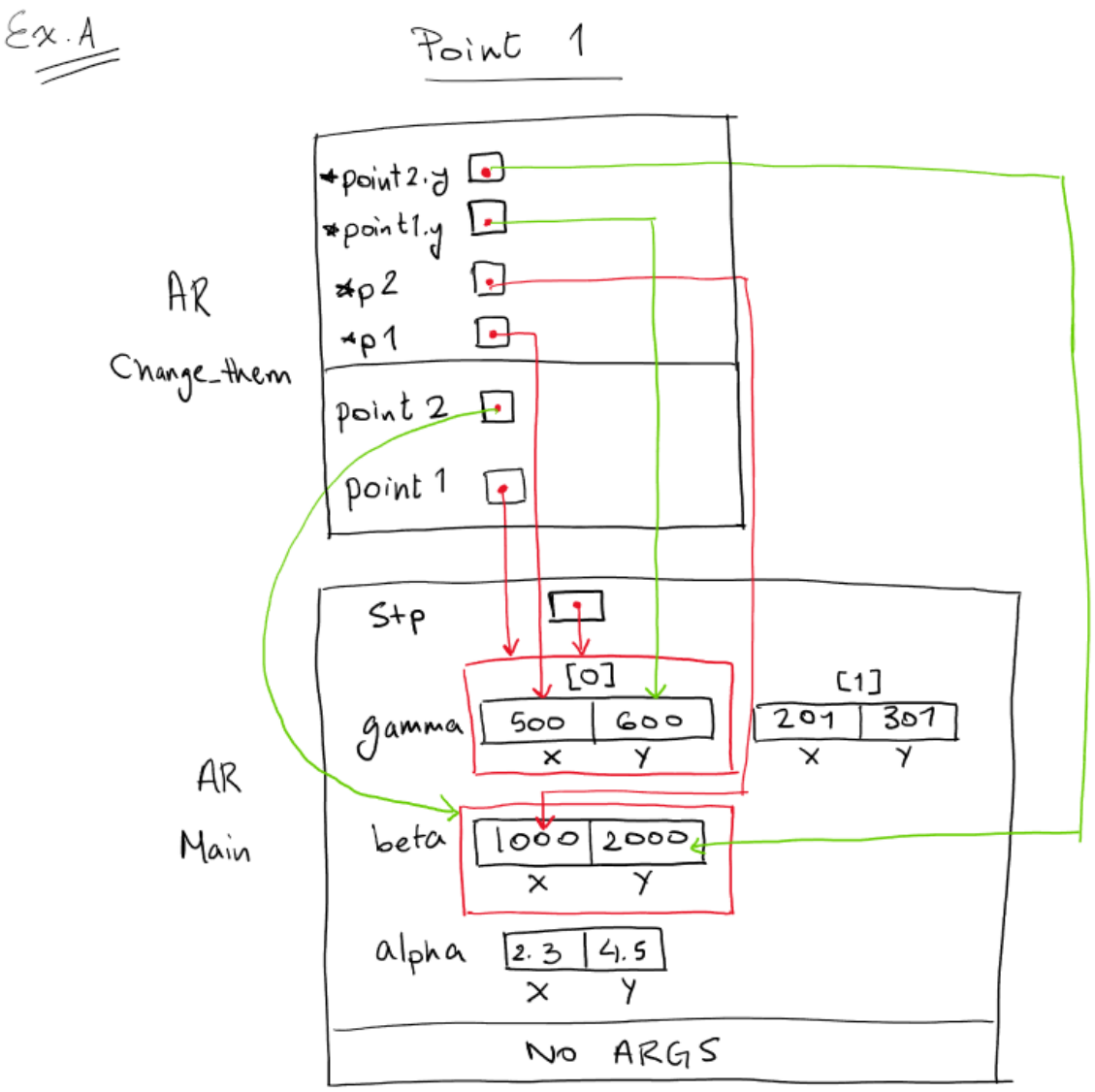
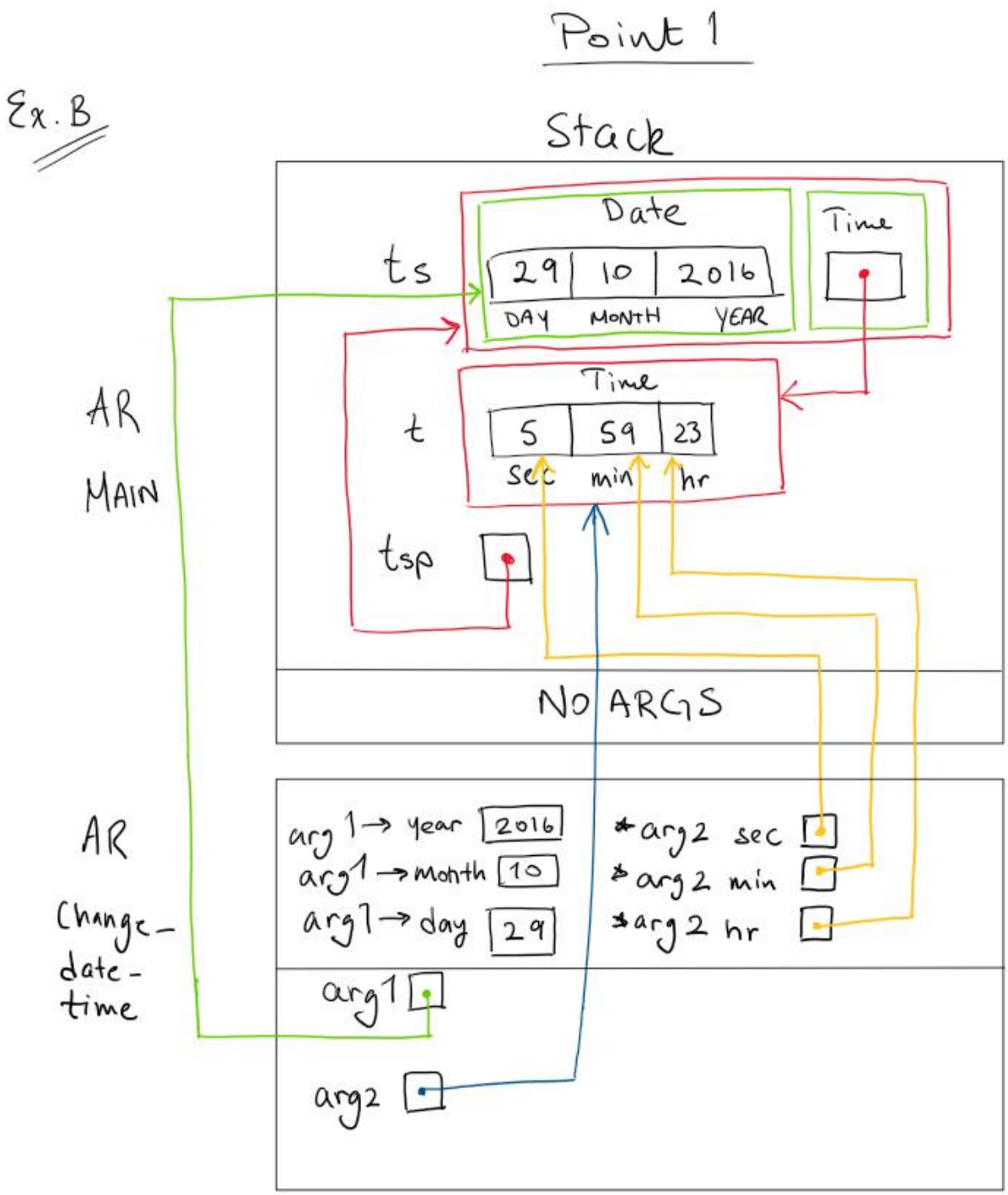
Course: Programming Fundamental – ENSF 337
Lab #: Lab 5
Instructor: M. Moussavi
Student Name: Aarushi Roy Choudhury
Lab Section: B01
Date submitted: Nov 1,2021

Exercise A


Ex. A
Point 1
AR
Change_them
*point2.y
*point1.y
*p2
*p1
point 2
point 1
Stp
[0]
gamma
500
600
x
y
[1]
201
301
x
y
AR
Main
beta
1000
2000
x
y
alpha
2.3
4.5
x
y
No ARGS

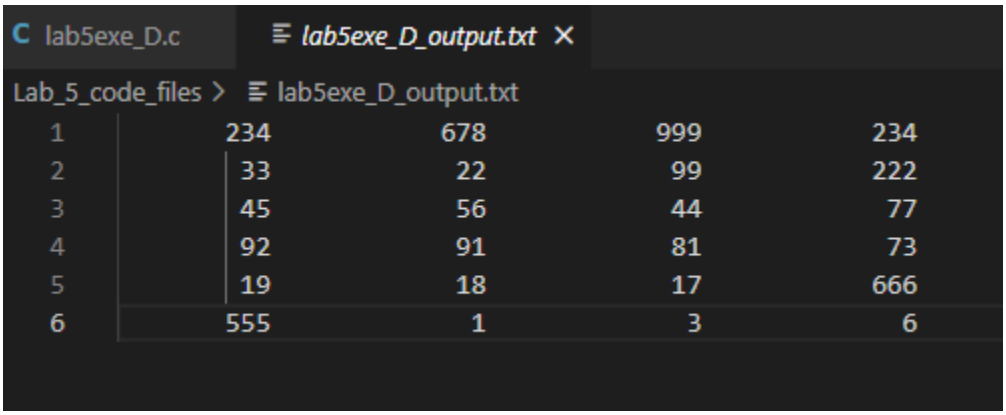
Exercise B


Point 1
Ex. B
Stack
Date
Time
ts
29
10
2016
DAY
MONTH
YEAR
Time
t
5
59
23
sec
min
hr
AR
MAIN
tsp
NO ARGS
AR
Change-date-time
arg1 → year 2016
arg1 → month 10
arg1 → day 29
*arg2 sec
*arg2 min
*arg2 hr
arg1
arg2

Exercise D

```c
#include <stdio.h>

#include <stdlib.h>

#include "lab5exe_D.h"

int main(void) {

    char input_filename[30] = "lab5exe_D.txt";

    char output_filename[30] = "lab5exe_D_output.txt";

    IntVector intVec;

    intVec.number_of_data = 0;

    read_text_file( & intVec, input_filename);

    display_single_column( & intVec);

    display_multiple_column( & intVec, 4, output_filename);

    return 0;

}

void read_text_file(IntVector * vec,
    const char * input_filename) {

    int nscan;

    FILE * fp = fopen(input_filename, "r");

    if (fp == NULL) {

        fprintf(stdout, "Sorry cannot open the text file %s.\n", input_filename);

        exit(1);

    }

    do {
```

```
        nscan = fscanf(fp, "%d", & vec -> storage[vec -> number_of_data]);

        if (nscan == 1)

            (vec -> number_of_data) ++;

        else if (nscan != EOF) {

            fprintf(stderr, "Invalid data in %s.\n", input_filename);

            exit(1);

        }

    } while ((nscan != EOF) & (vec -> number_of_data < MAX_CAPACITY));

    fclose(fp);

}

void display_single_column(const IntVector * intV) {

    int i;

    for (i = 0; i < intV -> number_of_data; i++)

        printf("%10d\n", intV -> storage[i]);

}

void display_multiple_column(const IntVector * intV, int col,const char *
output_filename)
{
        int i;
        FILE *fptr;

        fptr = fopen(output_filename,"w");

        for (i = 0; i < intV -> number_of_data; i++) {
                if(i>0 && i%col==0) fprintf(fptr,"\n");

        fprintf(fptr, "%10d\t", intV -> storage[i]);
        }
}
```


lab5exe_D.c
lab5exe_D_output.txt
Lab_5_code_files > lab5exe_D_output.txt
1 234 678 999 234
2 33 22 99 222
3 45 56 44 77
4 92 91 81 73
5 19 18 17 666
6 555 1 3 6

Exercise E

```c
// ENSF 337
//Lab5exe_E.c
//Completed By: Aarushi Roy Choudhury
#include "lab5exE.h"
#include <stdio.h>
#include <math.h>
#include <string.h>

int main(void){
    Point alpha = {"A1", 2.3, 4.5, 56.00};
    Point beta = {"B1", 25.9, 30.0, 97.00};
    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 ("\nExpected 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. (Expected to be:
53.74)", distance(&alpha, &beta));
    printf("\nThe distance between gamma and beta is: %.2f. (Expected to be:
26.87) \n", distance(&gamma, &beta));

    return 0;
```

```
}

void display_struct_point(const Point p){
    printf("\n%s <%.2lf, %.2lf, %.2lf>", p.label, p.x, p.y, p.z);
}


Point mid_point(const Point *p1, const Point *p2, const char *label){
    //YOU ARE NOT ALLOWED TO USE ANY STRING LIBRARY FUNCTIONS IN THIS FUNCTION
        //Create new point named label that is in the middle of p1 and p2

        int i;
        Point middle;

        double x=((*p1).x+(*p2).x)/2;
        double y=((*p1).y+(*p2).y)/2;
        double z=((*p1).z+(*p2).z)/2;

        for(i=0; label[i] != '\0'; i++){
                middle.label[i]=label[i];
        }
        middle.label[i]='\0';

        middle.x=x;
        middle.y=y;
        middle.z=z;

    return middle;
}
void swap(Point *p1, Point *p2){
        //Swaps the values of p1 and p2

        Point temp=*p1;
        *p1=*p2;
        *p2=temp;
}
double distance(const Point *p1, const Point *p2){
    //YOU ARE NOT ALLOWED TO USE THE ARROW OPERATOR (->)
        //Finds the distance between p1 and p2

        double d=sqrt(pow(((*p1).x-(*p2).x), 2) + pow(((*p1).y-(*p2).y), 2) +
pow(((*p1).z-(*p2).z), 2));

    return d;
}
```

```
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)
PS C:\Users\Aarus\Desktop\Lab_5>
```

Exercise F

```c
#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("Array of Points contains: \n");
    for(i=0; i < 10; i++){
        display_struct_point(struct_array[i], i);
        }

    printf("\nTesting the search function... \n");
    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 \n", position,
struct_array[position].label);
        }
    else{
        printf("\nstruct_array doesn't have label: %s. \n", "M1");
    }

    printf("\nTesting the reverse function... \n");
    reverse(struct_array, 10);
    printf("\nThe reversed array is: \n");
    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("struct_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 < n; 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* target, int n){
    //YOU ARE NOT ALLOWED TO USE ANY C LIBRARY FUNCTION IN YOUR SOLUTION
        //Returns index of first occurence of target in struct_array

        int i=0, j=0, index=0;

        if(struct_array[0].label[0]=='\0'){
                return -1;
        }

        while((struct_array[i].label[j] != '\0' || target[j] != '\0') &&
i<n){       //Cycles through struct_array to check if label matches target
                if(struct_array[i].label[j]==target[j]){
                        index=i;
                        j++;
```

```
                }
                else if((struct_array[i].label[j] != '\0' && target[j]=='\0') ||
(struct_array[i].label[j]=='\0' && target[j] != '\0') ||
(struct_array[i].label[j] != target[j])){
                        index=-1;
                        i++;
                }
        }
    return index;
}
void reverse (Point *a, int n){
    //Reverses elements of array a with length n

        Point temp[n];
        int i, j, k, l;

        if(a!=NULL){                            //if a is not empty...
                for(i=0; i < n; i++){        //Copying a into temp
                        temp[i].x=a[i].x;
                        temp[i].y=a[i].y;
                        temp[i].z=a[i].z;

                        for(j=0; a[i].label[j] != '\0'; j++){
                                temp[i].label[j]=a[i].label[j];
                        }
                        temp[i].label[j]='\0';
                }

                for(k=0; n >= 1; k++){              //Reversing a
                        a[k].x=temp[n-1].x;
                        a[k].y=temp[n-1].y;
                        a[k].z=temp[n-1].z;

                        for(l=0; a[k].label[l] != '\0'; l++){
                                a[k].label[l]=temp[n-1].label[l];
                        }
                        a[k].label[l]='\0';

                        n--;
                       }
                }
        else{
                printf("Array could not be reversed");
        }
}
```

```
PS C:\Users\Aarus\Desktop\Lab_5> cd "c:\Users\Aarus\Desktop\Lab_5\" ; if ($?) { gcc lab5exF.c -o
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>

Testing 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>
PS C:\Users\Aarus\Desktop\Lab_5>
```