Course: Programming Fundamental – ENSF337

Lab #: Lab 3

Instructor: M. Moussavi

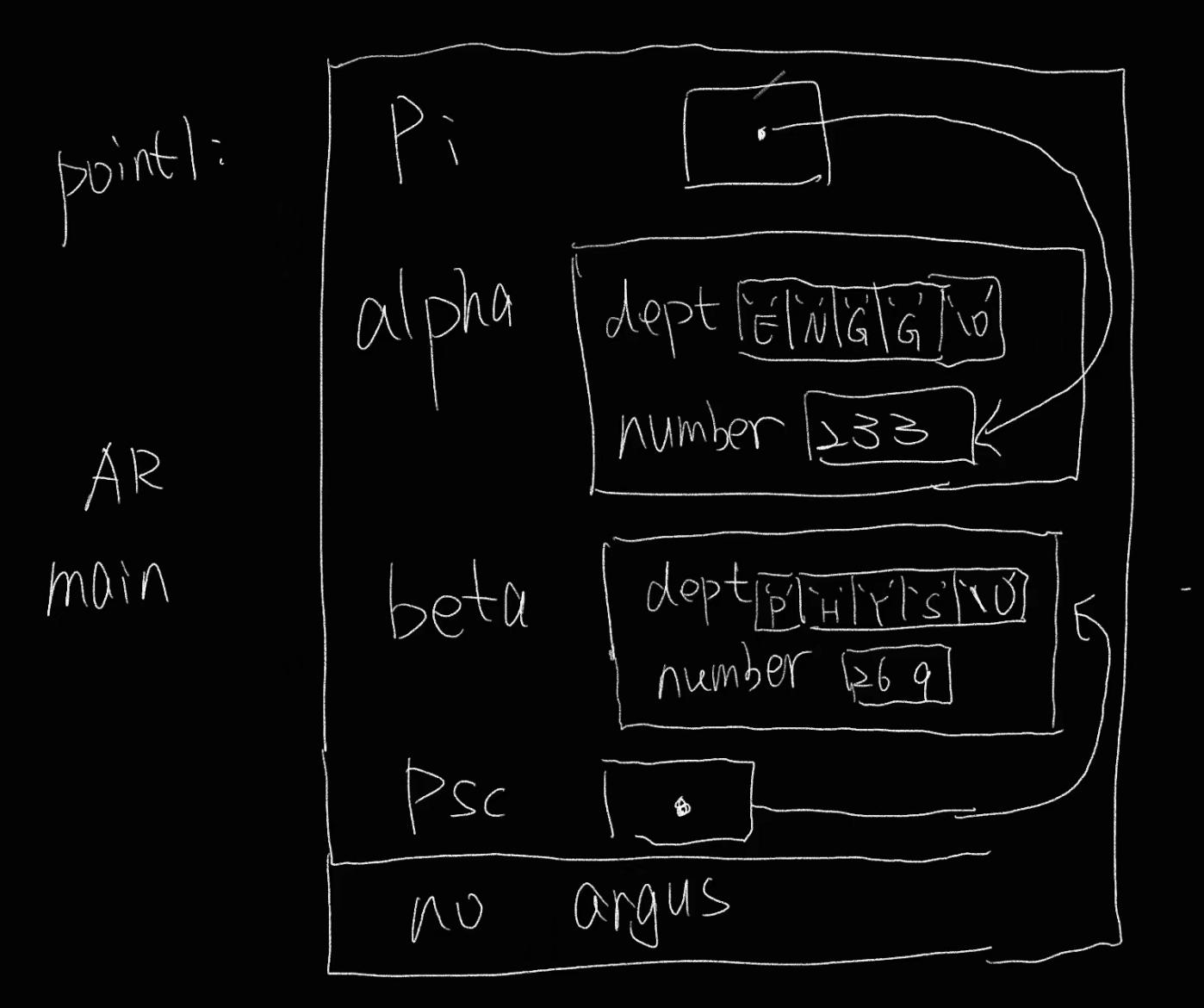
Student Name: Shanzi Ye

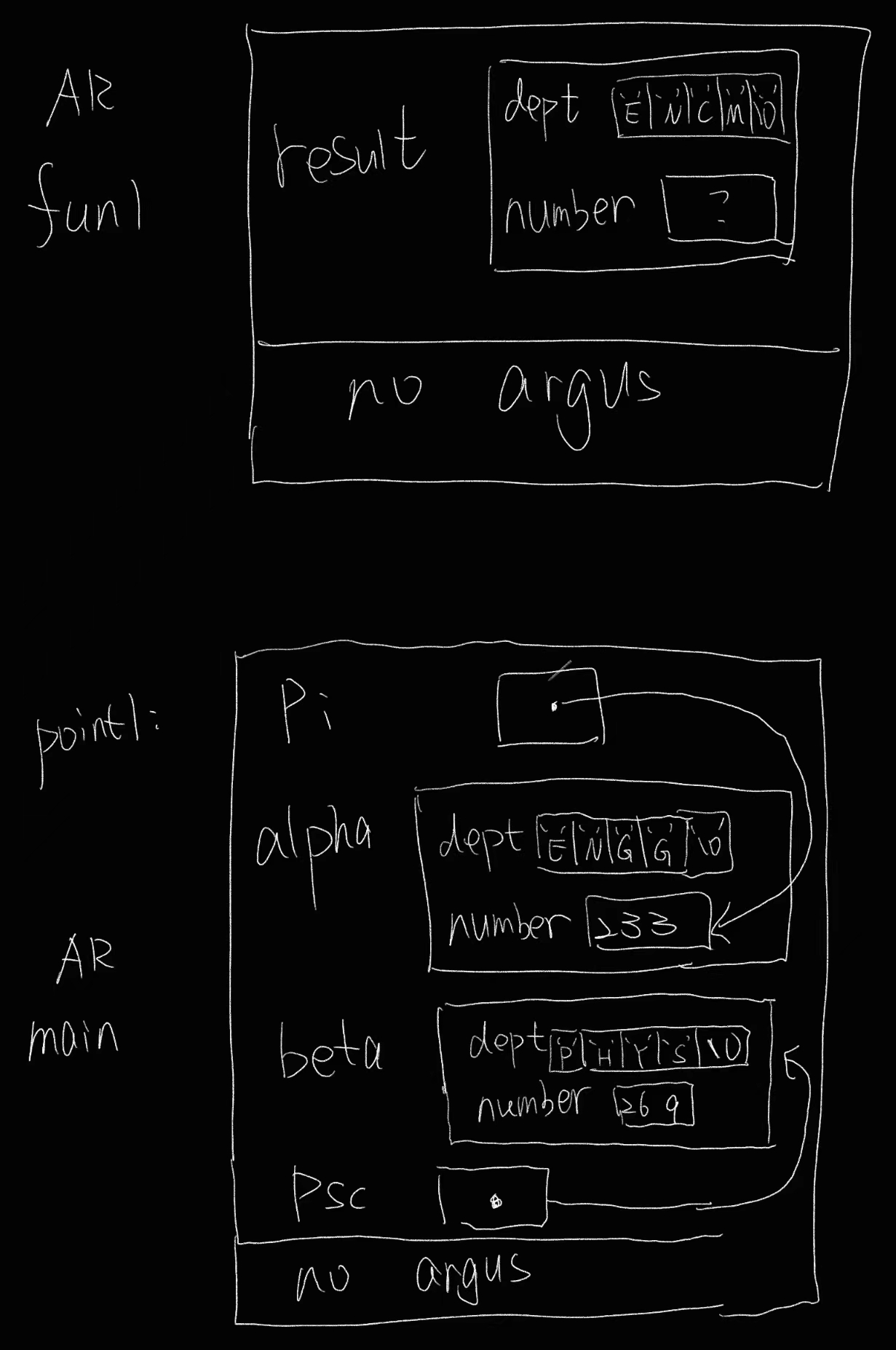
Lab Section: B01

Date submitted: May 24, 2022

# Exercise A

Point 1：



Point 2:

# Exercise B

struct cplx

cplx\_add(struct cplx z1, struct cplx z2)

{

struct cplx result;

result.real = z1.real + z2.real;

result.imag = z1.imag + z2.imag;

return result;

}

void cplx\_subtract(struct cplx z1, struct cplx z2, struct cplx\* difference)

{

struct cplx result;

result.real = z1.real - z2.real;

result.imag = z1.imag - z2.imag;

\*difference = result;

}

void cplx\_multiply(const struct cplx\* pz1, const struct cplx\* pz2, struct cplx\* product)

{

struct cplx result;

result.real = pz1->real \* pz2->real;

result.imag = pz1->imag \* pz2->imag;

\*product = result;

}

**You can check out the complete code in the file called “source code”.**

**I only wrote down the three definitions in the PDF.**

# Exercise C

/\*

\* File Name: lab3\_exe\_B.c

\* Assignment: Lab 3 Exercise B

\* Lab section: (B01)

\* Completed by: Shanzi Ye

\* Submission Date: May 24,2022

\*/

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include<math.h>

const int ARRAY\_SIZE = 10;

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

struct point

{

char label[3]; // 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

};

void display\_struct\_point(struct point x);

double distance (const struct point\* a, const struct point\* b);

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

int main(void)

{

struct point alpha = { "A1", 2.3, 4.5} ;

struct point \*stp = &alpha;

printf("Size of struct-point in our Linux lab is: %d bytes.\n",

(int) sizeof(struct point));

printf("Size of strcut-point pointer in our Linux lab is: %d bytes.\n",

(int) sizeof(stp));

printf("Size of strcut that stp points to is: %d bytes.\n",

(int) sizeof(\*stp));

display\_struct\_point(\*stp);

struct point sigma = { "C1", 12.3, 14.5, 56.00 } ;

struct point omega = { "D1", 125.9, 130.0, 97.00 } ;

struct point theta = { "E1", 5.9, 303.0, 7.00 } ;

display\_struct\_point(sigma);

display\_struct\_point(omega);

printf("\nThe distance between sigma and omega is: %10.2f", distance(&sigma, &omega));

printf("\nThe distance between sigma and theta is: %10.2f", distance(&sigma, &theta));

system("pause");

return 0;

}

void display\_struct\_point(struct point x)

{

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

}

double distance(const struct point\* p1, const struct point\* p2)

{

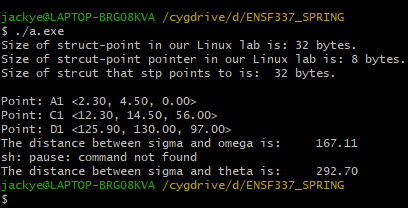
double distance\_3d = sqrt(pow((p1->x - p2->x), 2) + pow((p1->y - p2->y), 2) + pow((p1->z - p2->z), 2));

return distance\_3d;

/\*This funciton is incomplete and needs to be changed and complted by

the students to calculate and return the distance betwn the two three-D points\*/

}



# Exercise D

/\*

\* File Name: lab3\_exe\_D.c

\* Assignment: Lab 3 Exercise D

\* Lab section: (B01)

\* Completed by: Shanzi Ye

\* Submission Date: May 24,2022

\*/

#include <stdio.h>

#include <string.h>

const int ARRAY\_SIZE = 10;

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

struct point

{

char label[3]; // 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

};

void display\_struct\_point(struct point x);

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

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

int main(void)

{

struct point struct\_array[ARRAY\_SIZE];

int i;

int position;

populate\_struct\_array(struct\_array, ARRAY\_SIZE);

printf("\nArray of point structures contains: \n");

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

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

printf("\nNow testing function search ...\n");

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

if(position != -1)

printf("\nstruct\_array[%d] contains %s", position,

struct\_array[position].label);

else

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

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

if(position != -1)

printf("\nstruct\_array[%d] contains %s", position,

struct\_array[position].label);

else

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

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

if(position != -1)

printf("\nstruct\_array[%d] contains %s", position,

struct\_array[position].label);

else

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

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

if(position != -1)

printf("\nstruct\_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("\nstruct\_array[%d] contains %s", position,

struct\_array[position].label);

else

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

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

if(position != -1)

printf("\nstruct\_array[%d] contains %s", position,

struct\_array[position].label);

else

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

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

if(position != -1)

printf("\nstruct\_array[%d] contains %s", position,

struct\_array[position].label);

else

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

return 0;

}

void display\_struct\_point(struct point x)

{

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

}

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

{

int noFindFlag = 0;

int i;

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

{

int j = 0;

while (label[j] != '\0')

{

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

{

noFindFlag = 1;

break;

}

else

{

noFindFlag = 0;

}

j++;

}

if (noFindFlag == 0)

{

return i;

}

}

return -1;

}

void populate\_struct\_array(struct 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

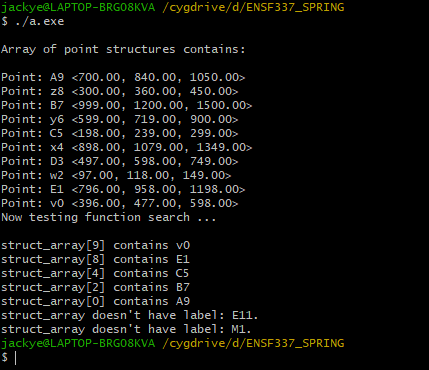
array[i].label[0] = ch3--;

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

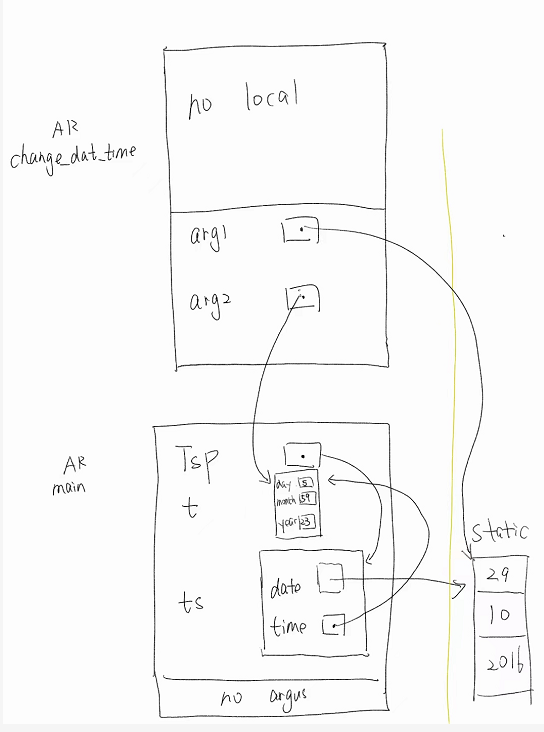
array[i].label[2] = '\0';

}

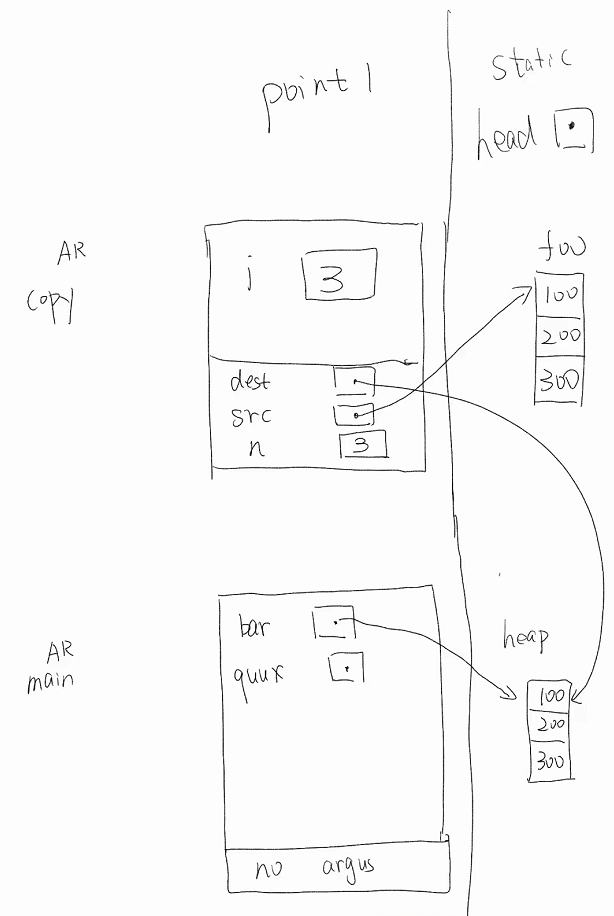
}



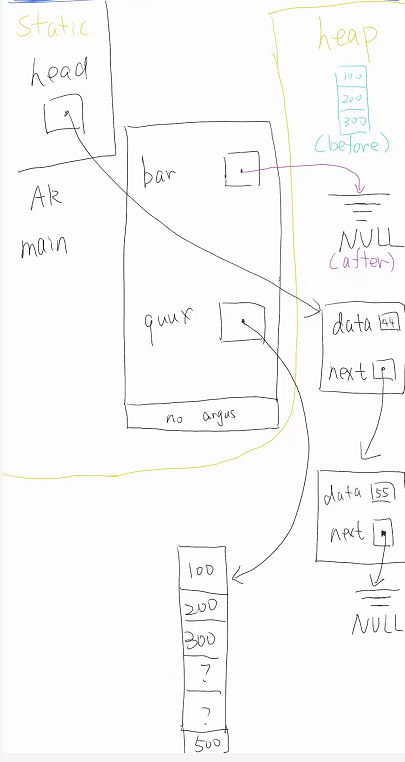
# Exercise E



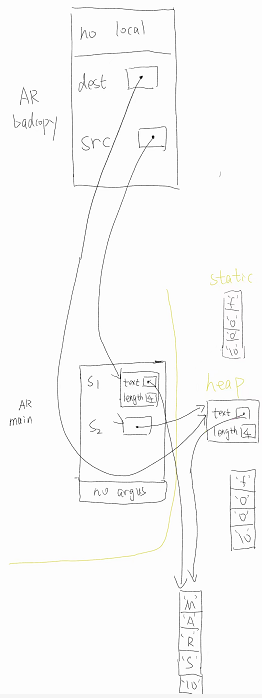
# Exercise F

Point 1:

Point 2:



# Exercise G



# Exercise I

/\*

\* File Name: lab3\_exe\_I.c

\* Assignment: Lab 3 Exercise I

\* Lab section: (B01)

\* Completed by: Shanzi Ye

\* Submission Date: May 24,2022

\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include "lab3exe\_I.h"

int main(void) {

char input\_filename[30] = "D:/lab3exe\_I.txt";

char output\_filename[30]= "lab3exe\_I\_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 a1 = intV->number\_of\_data / col;

int a2 = intV->number\_of\_data % col;

int i;

int j;

int arrayindex = 0;

for (i = 0; i < a1; i++)

{

for (j = 0; j < col; j++)

{

printf("%10d", intV->storage[arrayindex]);

arrayindex++;

}

printf("\n");

}

for (int k = 0; k<a2;k++)

{

printf("%10d",intV->storage[arrayindex]);

arrayindex++;

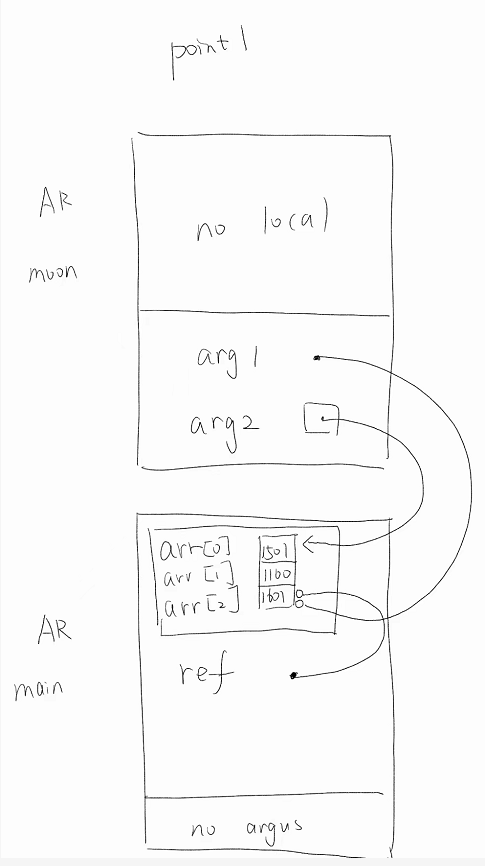
}

// STUDENTS MUST COMPLETE THE DEFINITION OF THIS FILE

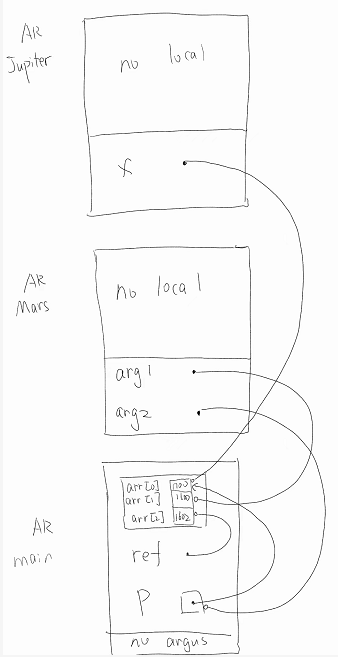
}

# Exercise J

Point1：



Point2：



//The picture is a little bit vague, arr[0] is 1700, arr[1] is 1100, arr[2] is 1602