Course: ENSF 694 – Summer 2024
Lab #: 1
Instructor: M. Moussavi
Student Name: Jaskirat Singh (Jazz)
Submission Date: 3 July 2024

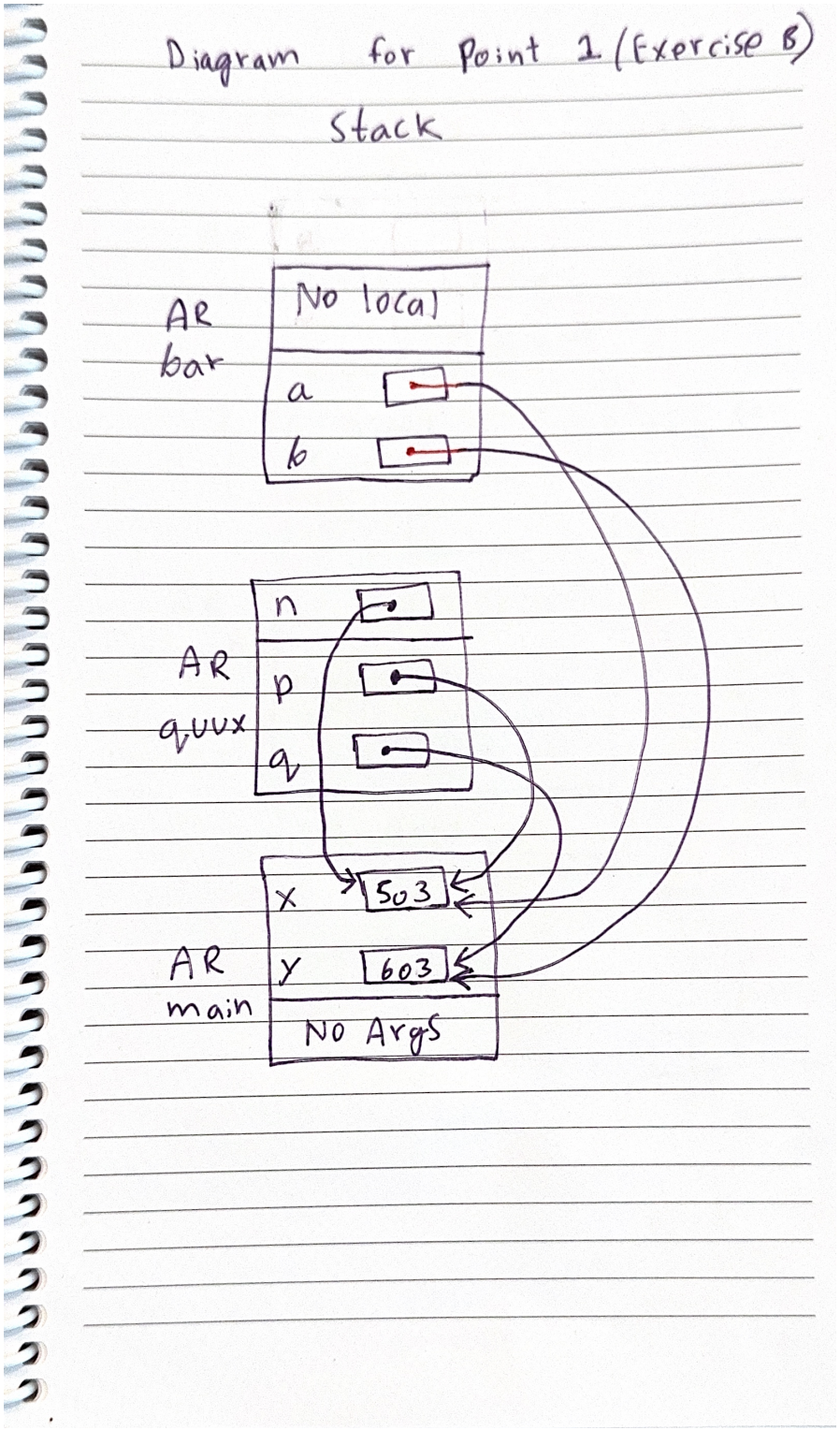
Exercise A Code:

```
/*
 *  lab1exe_A.cpp
 *  ENSF 694 Lab 1, exercise A
 *  Created by Mahmood Moussavi
 *  Completed by: Jaskirat Singh
 *  Submission date: July 3
 */

//Lines starting with # are a pre-processor directive
#include <iostream>
#include <cmath>
using namespace std;

const double G = 9.8;   /* gravitation acceleration 9.8 m/s^2 */
const double PI = 3.141592654;

void create_table(double v);
double Projectile_travel_time(double a, double v);
double Projectile_travel_distance(double a, double v);
double degree_to_radian(double d);

int main(void)
{
    double velocity;

    cout << "Please enter the velocity at which the projectile is launched (m/sec): ";
    cin >> velocity;

    if(!cin)  // means if cin failed to read
    {
        cout << "Invlid input. Bye...\n";
        exit(1);
    }

    while (velocity < 0 )
    {
        cout << "\nplease enter a positive number for velocity: ";
        cin >> velocity;
        if(!cin)
        {
            cout << "Invlid input. Bye...";
            exit(1);
        }
    }

    create_table(velocity);


    return 0;
}

void create_table(double v) {
    //Print header elements and units
    cout << "Angle \t t \t d" << endl;
    cout << "(deg) \t (sec) \t (m)" << endl;

    //Calculate and print variable values
    for(double deg = 0; deg <= 90; deg += 5) {
        cout << deg;
        cout << " \t ";
        cout << Projectile_travel_time(deg, v);
        cout << " \t ";
        cout << Projectile_travel_distance(deg, v) << endl;
    }
}

double Projectile_travel_time(double a, double v) {
    return 2 * v * sin(degree_to_radian(a)) / G;
}

double Projectile_travel_distance(double a, double v) {
    return v * v * sin(2 * degree_to_radian(a)) / G;
}

double degree_to_radian(double d) {
    return d * PI / 180;
}
```

Exercise A Output:

```
Please enter the velocity at which the projectile is launched (m/sec): 10
Angle    t   d
(deg)    (sec)   (m)
0    0   0
5    0.177869    1.77192
10   0.354384    3.49
15   0.528202    5.10204
20   0.698   6.55906
25   0.862486    7.81678
30   1.02041     8.83699
35   1.17056     9.5887
40   1.31181     10.0491
45   1.44308     10.2041
50   1.56336     10.0491
55   1.67174     9.5887
60   1.7674      8.83699
65   1.84961     7.81678
70   1.91774     6.55906
75   1.97128     5.10204
80   2.00981     3.49
85   2.03305     1.77192
90   2.04082     -4.18578e-09
Program ended with exit code: 0
```

Exercise B AR Diagram:



Exercise C Code:

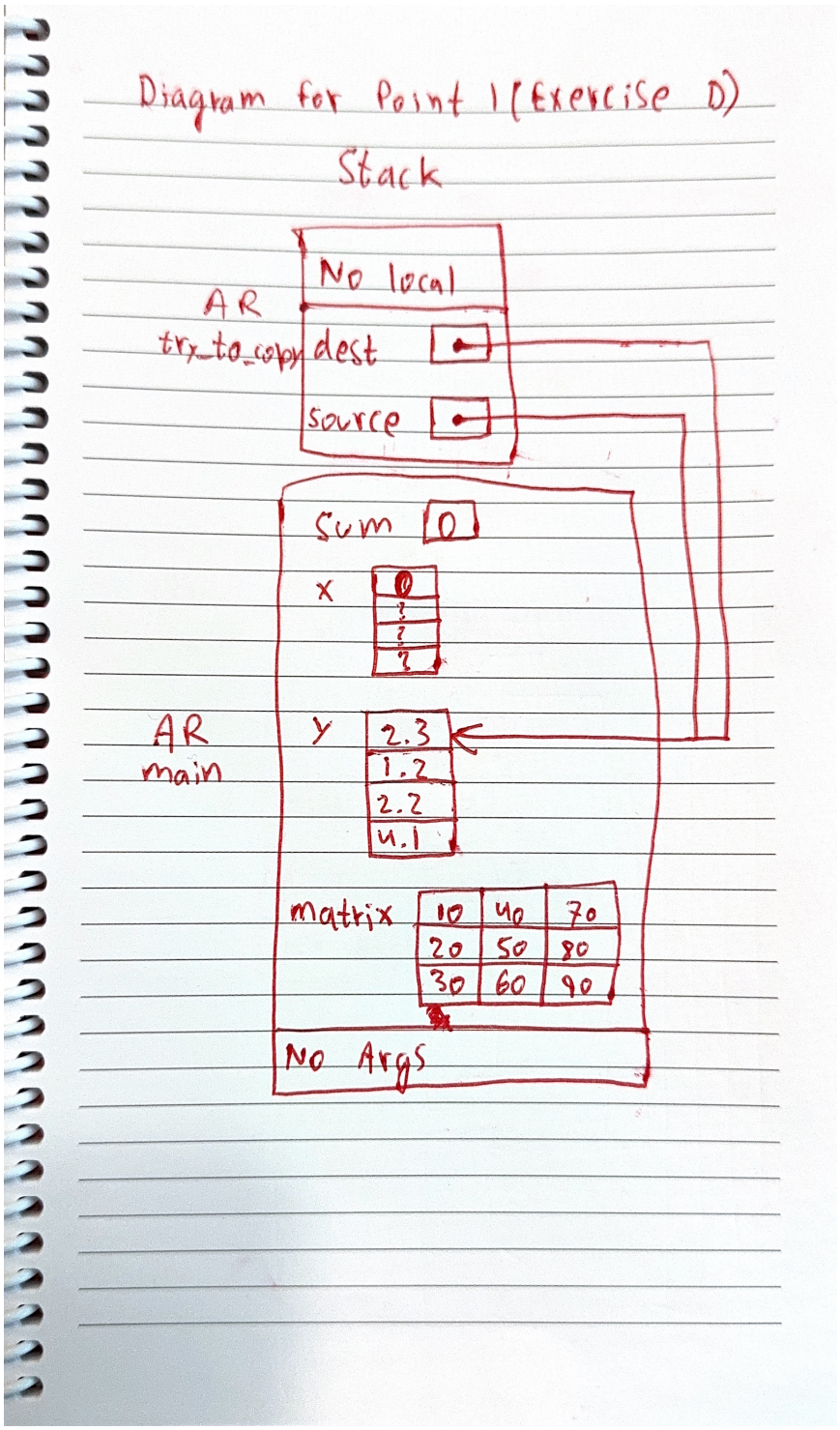
```
/*
 *  lab1exe_C.cpp
 *  ENSF 694 Lab 1, exercise C
 *  Completed by: Jaskirat Singh
 *  Submission date: July 3
 */

#include <iostream>
using namespace std;

void time_convert(int ms_time, int *minutes_ptr, double *seconds_ptr);
/*
 * Converts time in milliseconds to time in minutes and seconds.
 * For example, converts 123400 ms to 2 minutes and 3.4 seconds.
 * REQUIRES:
 *    ms_time >= 0.
 *    minutes_ptr and seconds_ptr point to variables.
 * PROMISES:
 *    0 <= *seconds_ptr & *seconds_ptr < 60.0
 *    *minutes_ptr minutes + *seconds_ptr seconds is equivalent to
 *    ms_time ms.
 */

int main(void)
{
  int millisec;
  int minutes;
  double seconds;

  cout << "Enter a time interval as an integer number of milliseconds: ";

 // printf("Enter a time interval as an integer number of milliseconds: ");
  cin >> millisec;

  if (!cin) {
    cout << "Unable to convert your input to an int.\n";
    exit(1);
  }

  cout << "Doing conversion for input of " <<  millisec <<" milliseconds ... \n";

  /* MAKE A CALL TO time_convert HERE. */
  time_convert(millisec, &minutes, &seconds);

  cout << "That is equivalent to " << minutes << " minute(s) and " << seconds << "
second(s).\n";
  return 0;
}

/* PUT YOUR FUNCTION DEFINITION FOR time_convert HERE. */
void time_convert(int ms_time, int *minutes_ptr, double *seconds_ptr) {
    //Populate seconds variable and minutes variable by reference
    *seconds_ptr = ms_time / 1000;
    *minutes_ptr = *seconds_ptr / 60;
    //Reduce the seconds variable to subtract the minutes contained in minutes
variable
    *seconds_ptr = fmod(*seconds_ptr, 60);
}
```

Exercise C Program output:

```
Enter a time interval as an integer number of milliseconds: 72000
Doing conversion for input of 72000 milliseconds ...
That is equivalent to 1 minute(s) and 12 second(s).
Program ended with exit code: 0
```

Exercise D AR Diagrams:

Point 1:



Point 2:

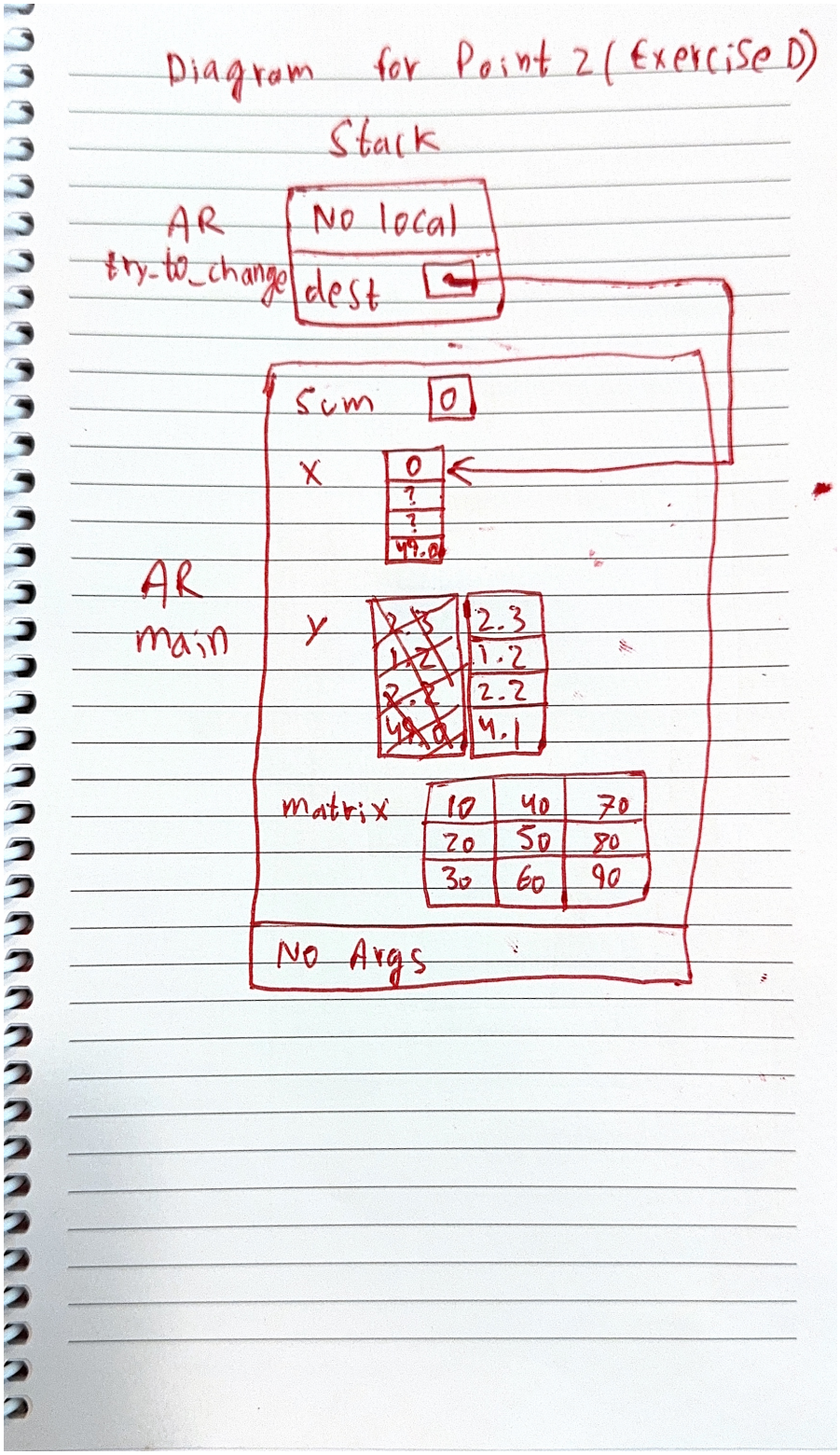


Diagram for Point 2 (Exercise D)
Stack
AR
try_to_change
No local
dest
Sum
0
x
0
?
?
49.0
AR
main
y
2.3
1.2
2.2
4.1
matrix
10
40
70
20
50
80
30
60
90
No Args

Point 3:

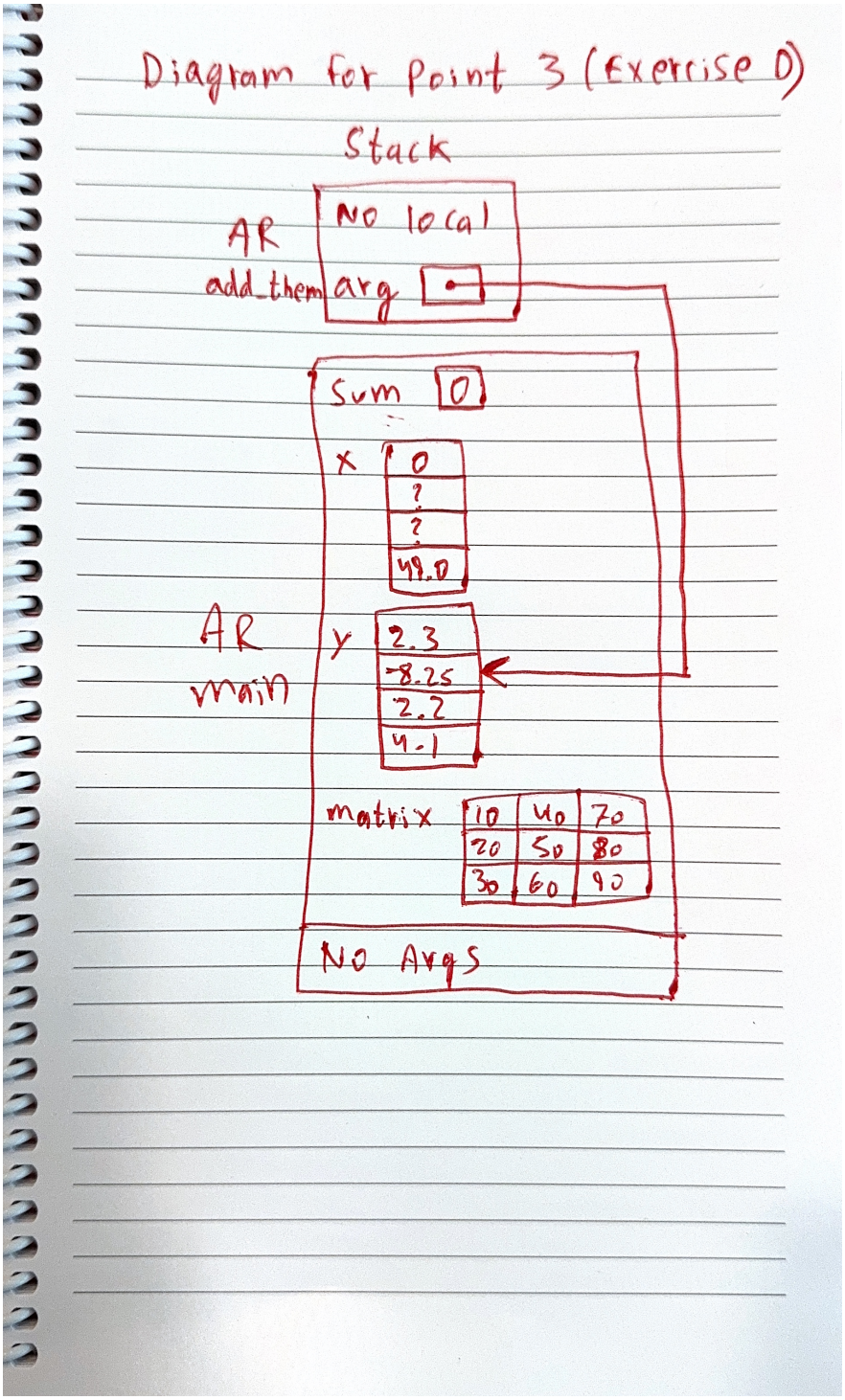


Diagram for Point 3 (Exercise 0)
Stack
AR
add_them
No local
arg
Sum
0
x
0
?
?
49.0
AR
main
y
2.3
-8.25
2.2
4.1
matrix
10
40
70
20
50
80
30
60
90
No Args

Exercise D Code:

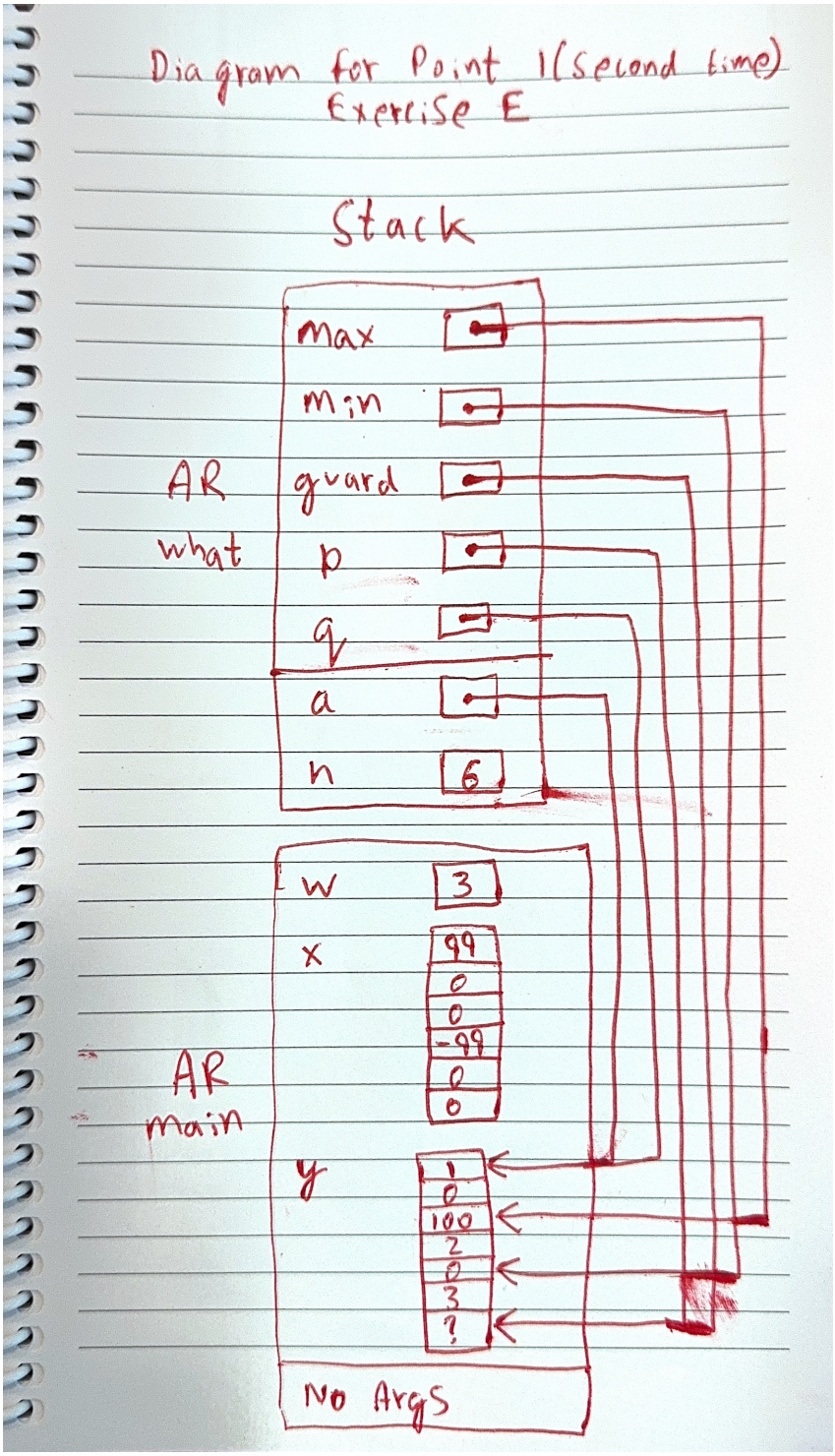
```
/*
 *  lab1exe_D.cpp
 *  ENSF 694 Lab 1, exercise D
 *  Completed by: Jaskirat Singh
 *  Submission date: July 3
 */

#include <iostream>
#include <iomanip>
using namespace std;
const int COL_SIZE = 3;
const int ROW_SIZE = 3;
void try_to_change(double* dest);
void try_to_copy(double dest[], double source[]);
double add_them (double a[5]);

void print_matrix(double matrix[][COL_SIZE], int rows);
/*
 * PROMISES: displays the values in the elements of the 2-D array, matrix,
 * formated in rows columns separated with one or more spaces.
 */

void good_copy(double *dest, double *source, int n);
/* REQUIRES: dest and source points to two array of double numbers with n to n-1 elements
 * PROMISES: copies the values in each element of array source to the corresponding element
 * in array dest.
 */

int main(void)
{
    double sum = 0;
    double x[4];
    double y[] = {2.3, 1.2, 2.2, 4.1};
    double matrix[ROW_SIZE][COL_SIZE] = { {10, 20, 30}, {40, 50, 60}, {70, 80, 90}};
    cout << " sizeof(double) is " << (int) sizeof(double) << " bytes.\n";
    cout << " size of x in main is: " << (int) sizeof(x) << " bytes.\n";
    cout << " y has " << (int) (sizeof(y)/ sizeof(double)) << " elements and its size is: " <<
(int) sizeof(y) << " bytes.\n";
    cout << " matrix has " << (int) (sizeof(matrix)/ sizeof(double)) << " elements and its size
is: " <<  (int) sizeof(matrix) << " bytes.\n";
    try_to_copy(x, y);
    try_to_change(x);

    sum = add_them(&y[1]);
    cout << "\n sum of values in y[1], y[2] and y[3] is: " << sum << endl;

    good_copy(x, y, 4);

    cout << "\nThe values in array x after call to good_copy are expected to be:";
    cout << "\n2.30, -8.25, 2.20, 4.10\n";
    cout << "And the values are:\n";
    for(int i = 0; i < 4; i++)
        cout << fixed << setprecision(2) << x[i] << "  ";

    cout << "\nThe values in matrix are:\n";
    print_matrix(matrix, 3);

    cout << "\nProgram Ends...\n";

    return 0;
}

void try_to_copy(double dest[], double source[])
{
    dest = source;
    /* point one*/
```

```
    return;
}

void try_to_change(double* dest)
{
    dest [3] = 49.0;

    /* point two*/
    cout << "\n sizeof(dest) in try_to_change is "<< (int)sizeof(dest) << " bytes.\n";
    return;
}


double add_them (double arg[5])
{
    *arg = -8.25;

    /* point three */
    cout << "\n sizeof(arg) in add_them is " << (int) sizeof(arg) << " bytes.\n";
    cout << "\n Incorrect array size computation: add_them says arg has " << (int) (sizeof(arg)/
sizeof(double)) <<" element.\n";

    return arg[0] + arg[1] + arg[2];
}

void good_copy(double *dest, double *source, int n)
{
    for(int i = 0; i < n; i++) {
        dest[i] = source[i];
    }
}

void print_matrix(double matrix[][COL_SIZE], int rows)
{
    for(int i = 0; i < rows; i++) {
        for(int j = 0; j < COL_SIZE; j++) {
            cout << matrix[i][j] << " ";
        }
        cout << "\n";
    }
}
```

Exercise D Program Output:

```
 sizeof(double) is 8 bytes.
 size of x in main is: 32 bytes.
 y has 4 elements and its size is: 32 bytes.
 matrix has 9 elements and its size is: 72 bytes.

 sizeof(dest) in try_to_change is 8 bytes.

 sizeof(arg) in add_them is 8 bytes.

 Incorrect array size computation: add_them says arg has 1 element.

 sum of values in y[1], y[2] and y[3] is: -1.95

The values in array x after call to good_copy are expected to be:
2.30, -8.25, 2.20, 4.10
And the values are:
2.30  -8.25  2.20  4.10
The values in matrix are:
10.00 20.00 30.00
40.00 50.00 60.00
70.00 80.00 90.00

Program Ends...
Program ended with exit code: 0
```

Exercise E AR Diagram:



Exercise F Code:

```
/*
 *  lab1exe_F.cpp
 *  ENSF 694 Lab 1, exercise F
 *  Completed by: Jaskirat Singh
 *  Submission date: July 3
 */

#include "MyArray.h"

int search(const MyArray* myArray, int obj){
// Students are supposed to complete the implementation of the this function
    for(int i = 0; i < myArray->list_size; i ++) {
        if(myArray->array[i] == obj) {
            return i;
        }
    }
    return -1;
}

void initialize(MyArray* myArray) {
    // Students are supposed to complete the implementation of the this function
    myArray->list_size = 0;
}

int retrieve_at(MyArray* myArray, int pos){
    // Students are supposed to complete the implementation of the this function
    return myArray->array[pos];
    return 0;
}

int count(MyArray* myArray, int obj ){
    // Students are supposed to complete the implementation of the this function
    int counter = 0;
    for(int i = 0; i < myArray->list_size; i ++){
        if(myArray->array[i] == obj) {
            counter ++;
        }
    }
    return counter;
}

void append( MyArray* myArray, int array[], int n ) {
    // Students are supposed to complete the implementation of the this function
    //Append array if condition is true, do nothing if it is false
    if(myArray->list_size + n <= SIZE) {
        //Append new array to existing array
        for(int i = 0; i < n; i ++) {
            myArray->array[myArray->list_size + i] = array[i];
        }
        //Increase array size
        myArray->list_size += n;
    }
}

void insert_at(MyArray* myArray, int pos, int val) {
    // Students are supposed to complete the implementation of the this function
    myArray->list_size ++;
    // Starting from the index of the new list size, move each element after pos to
the right
    for(int i = myArray->list_size - 1; i > pos; i --) {
```

```
        myArray->array[i] = myArray->array[i - 1];
    }
    // Insert new value into the freshly emptied space
    myArray->array[pos] = val;
}

int remove_at(MyArray* myArray, int pos ) {
    // Students are supposed to complete the implementation of the this function
    int valueRemoved = myArray->array[pos];
    for(int i = pos; i < myArray->list_size - 1; i ++) {
        myArray->array[i] = myArray->array[i+1];
    }
    //Decriment list_size to reflect new value
    myArray->list_size --;
    return valueRemoved;
}

int remove_all(MyArray* myArray, int value ) {
    // Students are supposed to complete the implementation of the this function
    int valueRemoved = -1;
    for(int i = 0; i < myArray->list_size; i ++) {
        if(myArray->array[i] == value) {
            valueRemoved = remove_at(myArray, i);
        }
    }
    return valueRemoved;
}

// You can modify this function however you want:  it will not be tested

void display_all(MyArray* myArray) {
    // Students are supposed to complete the implementation of the this function
    for(int i = 0; i < myArray->list_size; i ++) {
        cout << "Element " << i + 1 << ": " << myArray->array[i] << endl;
    }
}

bool is_full(MyArray* myArray){
    // Students are supposed to complete the implementation of the this function
    return myArray->list_size == SIZE;
}

bool isEmpty(MyArray* myArray){
    // Students are supposed to complete the implementation of the this function
    return myArray->list_size == 0;
}

int size(MyArray* myArray){
    // Students are supposed to complete the implementation of the this function
    return myArray->list_size;
}
```

Exercise F Program Output:

```
aether@Mac MyArrayFiles % g++ -Wall MyArray.cpp MyArray_tester.cpp -o myProgram
aether@Mac MyArrayFiles % ./myProgram
Starting Test Run. Using input file.
Line 1 >>  Passed
Line 2 >>  Passed
Line 3 >>  Passed
Line 4 >>  Passed
Line 5 >>  Passed
Line 6 >>  Passed
Line 7 >>  Passed
Line 8 >>  Passed
Line 9 >>  Passed
Line 10 >>  Passed
Line 11 >>  Passed
Line 12 >>  Passed
Line 13 >>  Passed
Line 14 >>  Passed
Line 15 >>  Passed
Line 16 >>  Passed
Line 17 >>  Passed
Line 18 >>  Passed
Line 19 >>  Passed
Exiting...
Finishing Test Run
Showing Data in the List:
101  200  100  500
Program Ended ....
```