Course: ENSF 694 – Summer 2024

Lab #: Lab 1

Instructor: Mahmood Moussavi

Student Name: Ryan Baker

Submission Date: June 27, 2024

EXERCISE A

lab1exe_A.cpp

```
/*
 *  lab1exe_A.cpp
 *  ENSF 694 Lab 1, exercise A
 *  Created by Mahmood Moussavi
 *  Completed by: Ryan Baker
 *  Development Date: June 26, 2024
 */

#include <iostream>
#include <cmath>
#include <iomanip> // included for table formatting
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);
/* REQUIRES
 *   v (double) the velocity to base the table on
 * PROMISES
 *   prints a table of angle (0-90 degrees) with corresponding times and distances
 */

double Projectile_travel_time(double a, double v);
/* REQUIRES
 *   0 < a < 90.
 * PROMISES
 *  return value is projectile travel time when maximum horizontal distance is reached (in
seconds).
 */

double Projectile_travel_distance(double a, double v);
/* REQUIRES
 *   0 < a < 90.
 * PROMISES
 *  return value is maximum projectile distance (in metres).
 */

double degree_to_radian(double d);
/* REQUIRES
 *  Angle in degrees between 0 and 360.
 * PROMISES
```

```
 *  returns value is angle in radians.
 */

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){
    cout << "Angle\t\t\tt\t\td\n(deg)\t\t(sec)\t\t(m)\n";
    cout << std::fixed << std::setprecision(6); // format digits of table
    for(double i = 0; i <= 90; i +=5){
        cout << i << "\t\t" << Projectile_travel_time(i, v) << "\t\t" <<
Projectile_travel_distance(i, v) << "\n";
    }
}


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

```
}


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


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

Code output:

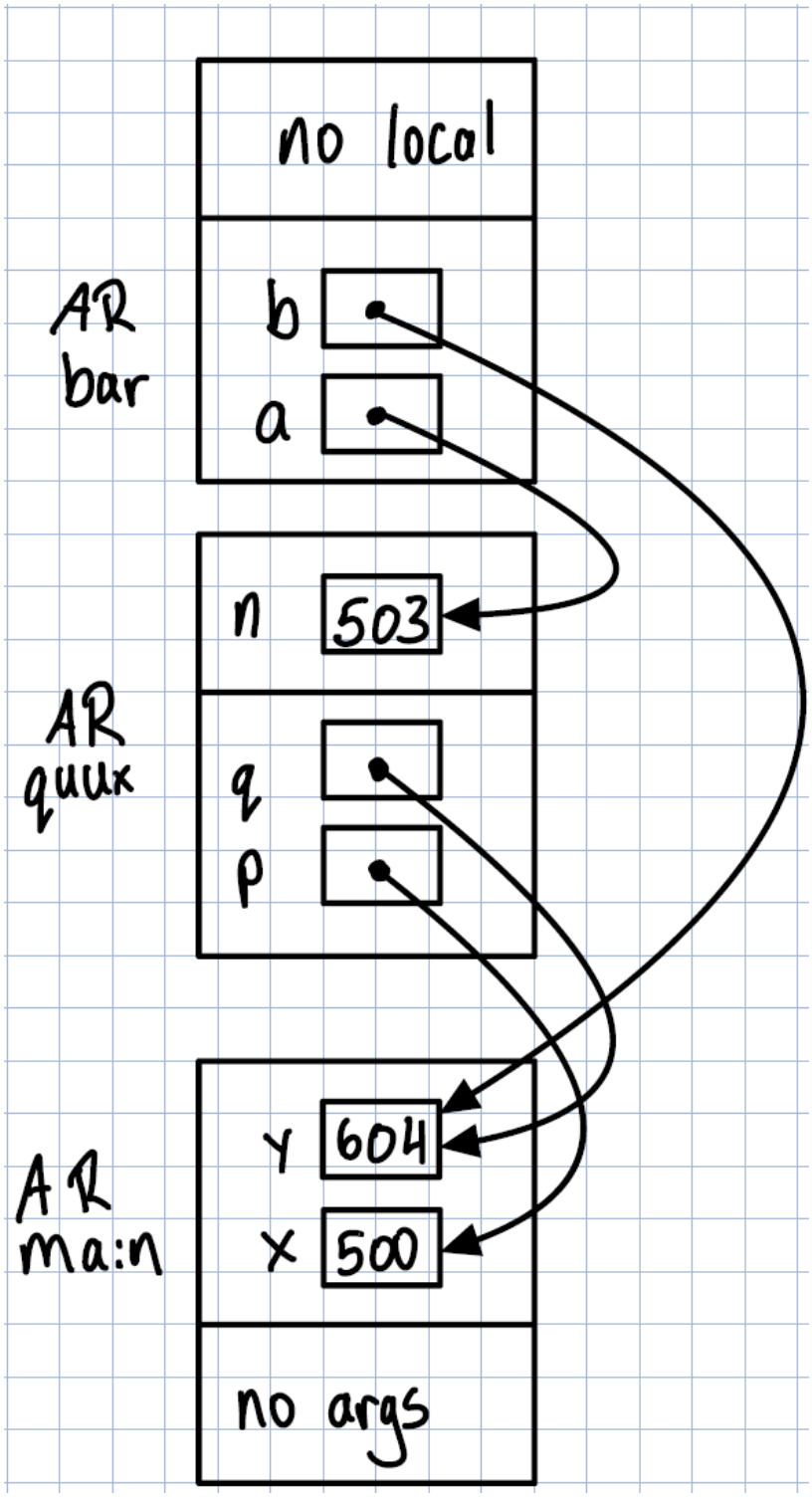
```
QWE+RyanB@ryanb-pc /cygdrive/c/Users/RyanB/OneDrive - Quick Way Electrical (1999) Ltd/ENSF
694/Labs/ensf-694-a01
$ ./lab1exe_A
Please enter the velocity at which the projectile is launched (m/sec): 100
Angle                   t                       d
(deg)                   (sec)                   (m)
0.000000                0.000000                0.000000
5.000000                1.778689                177.192018
10.000000               3.543840                349.000146
15.000000               5.282021                510.204082
20.000000               6.980003                655.905724
25.000000               8.624862                781.678003
30.000000               10.204082               883.699392
35.000000               11.705642               958.870021
40.000000               13.118114               1004.905870
45.000000               14.430751               1020.408163
50.000000               15.633560               1004.905870
55.000000               16.717389               958.870021
60.000000               17.673988               883.699391
65.000000               18.496077               781.678003
70.000000               19.177400               655.905724
75.000000               19.712772               510.204081
80.000000               20.098117               349.000146
85.000000               20.330504               177.192018
90.000000               20.408163               0.000000
```

EXERCISE B

Part I

- No submission required

Part II



EXERCISE C

lab1exe_C.cpp

```cpp
/*
 *  lab1exe_C.cpp
 *  ENSF 694 Lab 1 Exercise C
 *  Completed by: Ryan Baker
 *  Development Date: June 29, 2024
 */

#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){
  *minutes_ptr = ms_time/6000;
  *seconds_ptr = (double)(ms_time % 6000) / 1000.0;
}
```

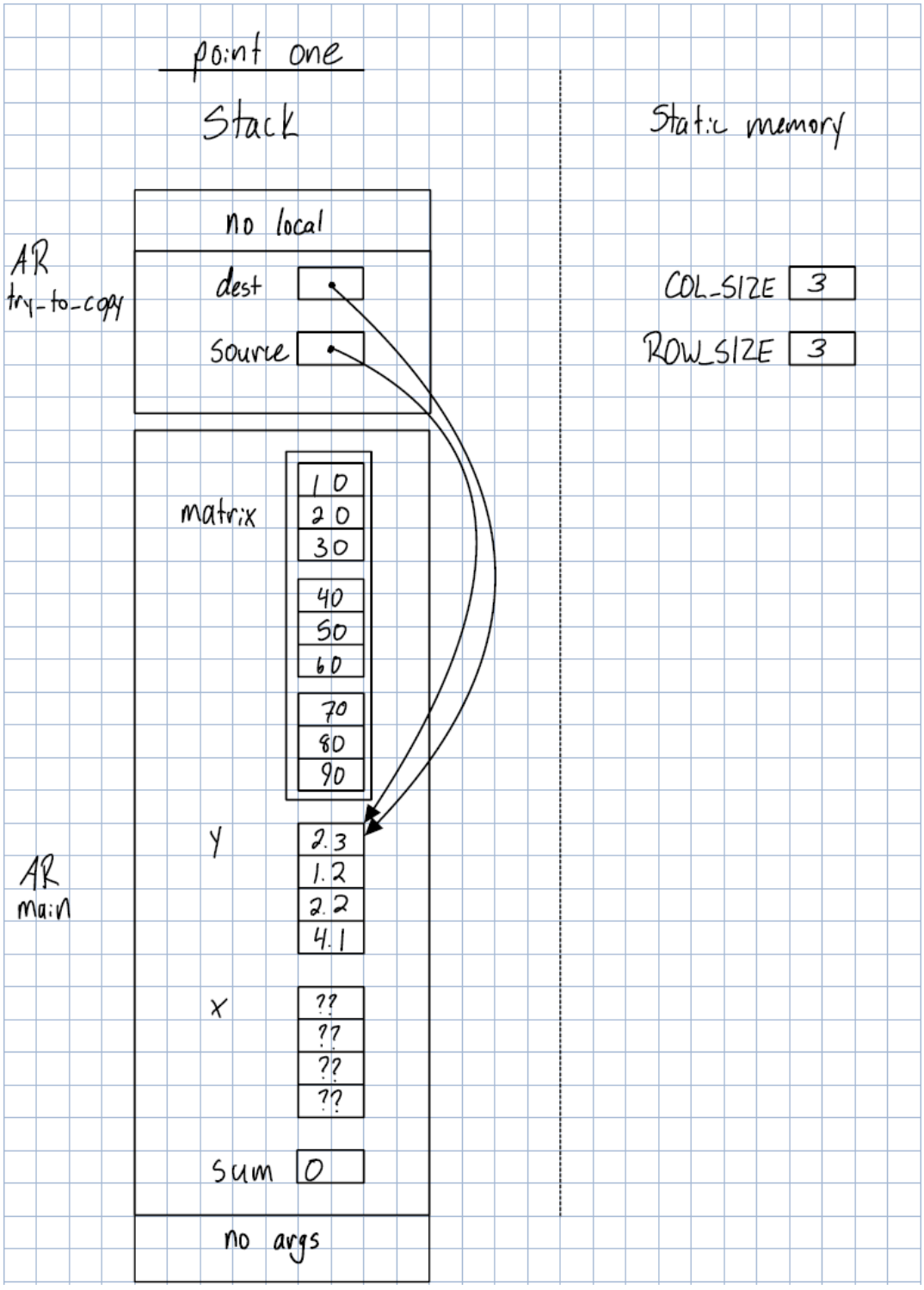
lab1exe_C.cpp output

```
QWE+RyanB@ryanb-pc /cygdrive/c/Users/RyanB/OneDrive - Quick Way Electrical (1999) Ltd/ENSF
694/Labs/ensf-694-a01
$ ./lab1exe_C
Enter a time interval as an integer number of milliseconds: 7450
Doing conversion for input of 7450 milliseconds ...
That is equivalent to 1 minute(s) and 1.45 second(s).
```

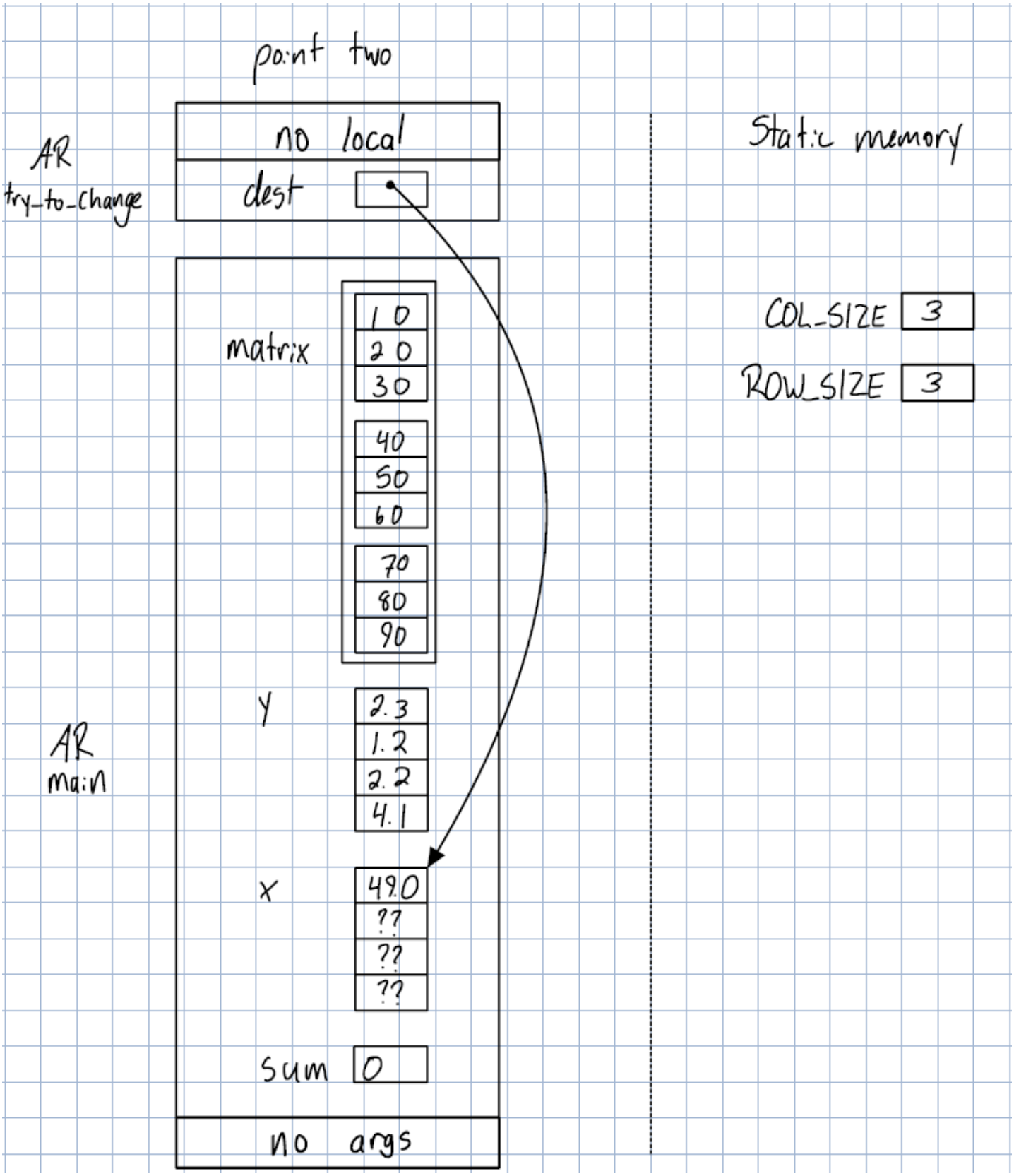
EXERCISE D

Part I

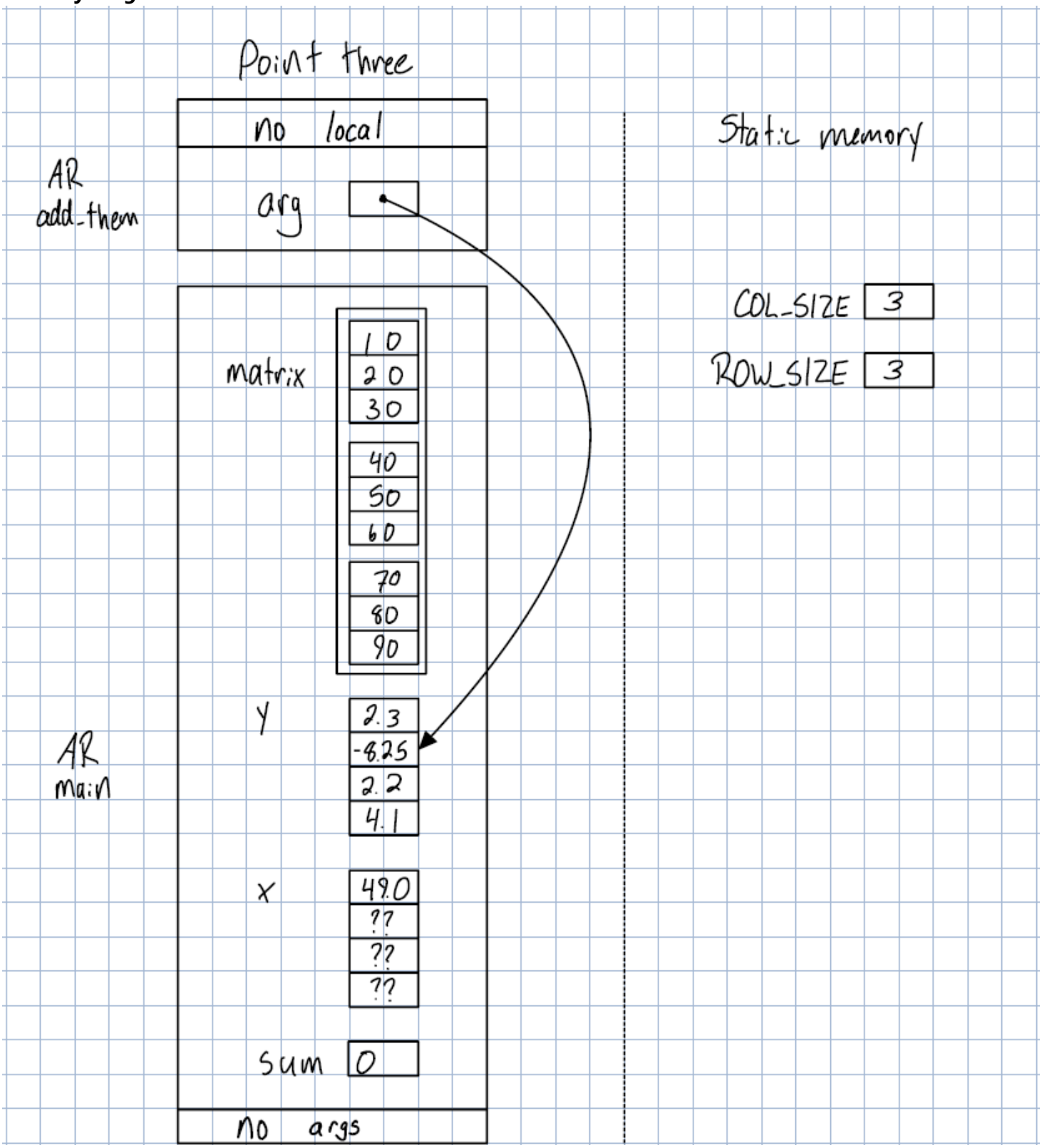
Memory Diagram – Point One



Memory Diagram – Point Two



Memory Diagram – Point Three



Part II

lab1exe_D.cpp

```
/*/
 *  lab1exe_D.cpp
 *  ENSF 694 Lab 1 Exercise D
 *  Completed by: Ryan Baker
 *  Development Date: June 29, 2024
 */

#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);
```

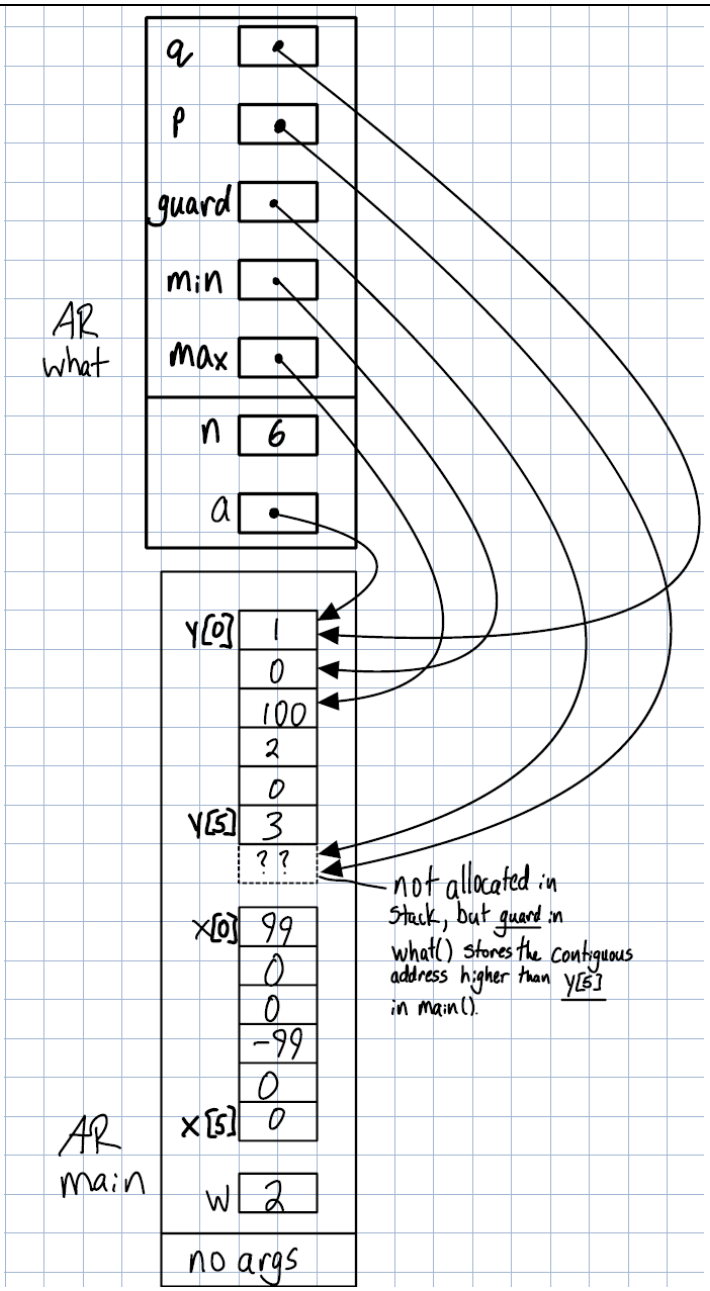
```
    sum = add_them(&y[1]);
    cout << "\n sum of values in y[1], y[2] and y[3] is: " << sum << endl;

    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 << endl;
    }
}
```

Output

```
QWE+RyanB@ryanb-pc /cygdrive/c/Users/RyanB/OneDrive - Quick Way Electrical (1999) Ltd/ENSF 694/Labs/ensf-694-a01
$ g++ -w lab1exe_D.cpp -o lab1exe_D

QWE+RyanB@ryanb-pc /cygdrive/c/Users/RyanB/OneDrive - Quick Way Electrical (1999) Ltd/ENSF 694/Labs/ensf-694-a01
$ ./lab1exe_D
 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...
```

EXERCISE E


q
p
guard
min
max
AR
what
n
6
a
y[0]
1
0
100
2
0
y[5]
3
??
not allocated in stack, but guard in what() stores the contiguous address higher than y[5] in main().
x[0]
99
0
0
-99
0
x[5]
0
AR
main
w
2
no args

EXERCISE F

MyArray.cpp

```cpp
/*
 *  MyArray.cpp
 *  ENSF 694 Lab 1 Exercise F
 *  Completed by: Ryan Baker
 *  Development Date: July 2, 2024
 */

#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];
}

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

void append( MyArray* myArray, int array[], int n ) {
    // Students are supposed to complete the implementation of the this function
    if((myArray->list_size + n) <= (int)(sizeof(myArray->array)/sizeof(int))){
        for(int i = 0; i < n; i++){
```

```
            myArray->array[myArray->list_size] = array[i];
            myArray->list_size++;
        }
    }
}

void insert_at(MyArray* myArray, int pos, int val) {
    // Students are supposed to complete the implementation of the this function
    myArray->list_size++;
    for(int i = myArray->list_size - 1; i > pos; i--){
        myArray->array[i] = myArray->array[i-1];
    }
    myArray->array[pos] = val;
}

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

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

// 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 << myArray->array[i] << " ";
    }
    cout << endl;
}
```

```
bool is_full(MyArray* myArray){
    // Students are supposed to complete the implementation of the this function
    return ((sizeof(myArray->array) / sizeof(int)) == myArray->list_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;
}
```

output.txt

```
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 ....
```