Course: ENSF694 – Summer 2025
Lab #: Lab 1
Instructor: Mahmood Moussavi
Student Name: John Zhou
Submission Date: July 10, 2025

Exercise A

```
/*
 *  lab1exe_A.cpp
 *  ENSF 694 Lab 1, exercise A
 *  Created by Mahmood Moussavi
 *  Completed by: John Zhou
 */

#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);
/*
 * REQUIRES:
 *   - velocity 'v' > 0.
 *
 * PROMISES:
 *   - Prints a table showing the time of flight and distance for projectile for each angle from 0° to 90°, at the given velocity.
 */
double Projectile_travel_time(double a, double v);
/*
 * REQUIRES:
 *   - a (angle) in degrees
```

```
 *   - v (velocity) in m/s (v > 0).
 *
 * PROMISES:
 *   - Returns the time of flight for the projectile
 *     Formula: time = (v^2 * sin(2 * a)) / g
 */
double Projectile_travel_distance(double a, double v);
/*
 * REQUIRES:
 *   - a (angle) in degrees
 *   - v (velocity) in m/s (v > 0).
 *
 * PROMISES:
 *   - Returns the horizontal distance the projectile will travel
 *     Formula: distance = (2 * v * sin(a)) / g
 */
double degree_to_radian(double d);
/*
 * REQUIRES:
 *   - angle 'd' in degrees
 *
 * PROMISES:
 *   - Returns the corresponding 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;
}

double degree_to_radian(double d)

{

    return d * PI / 180.0;
```

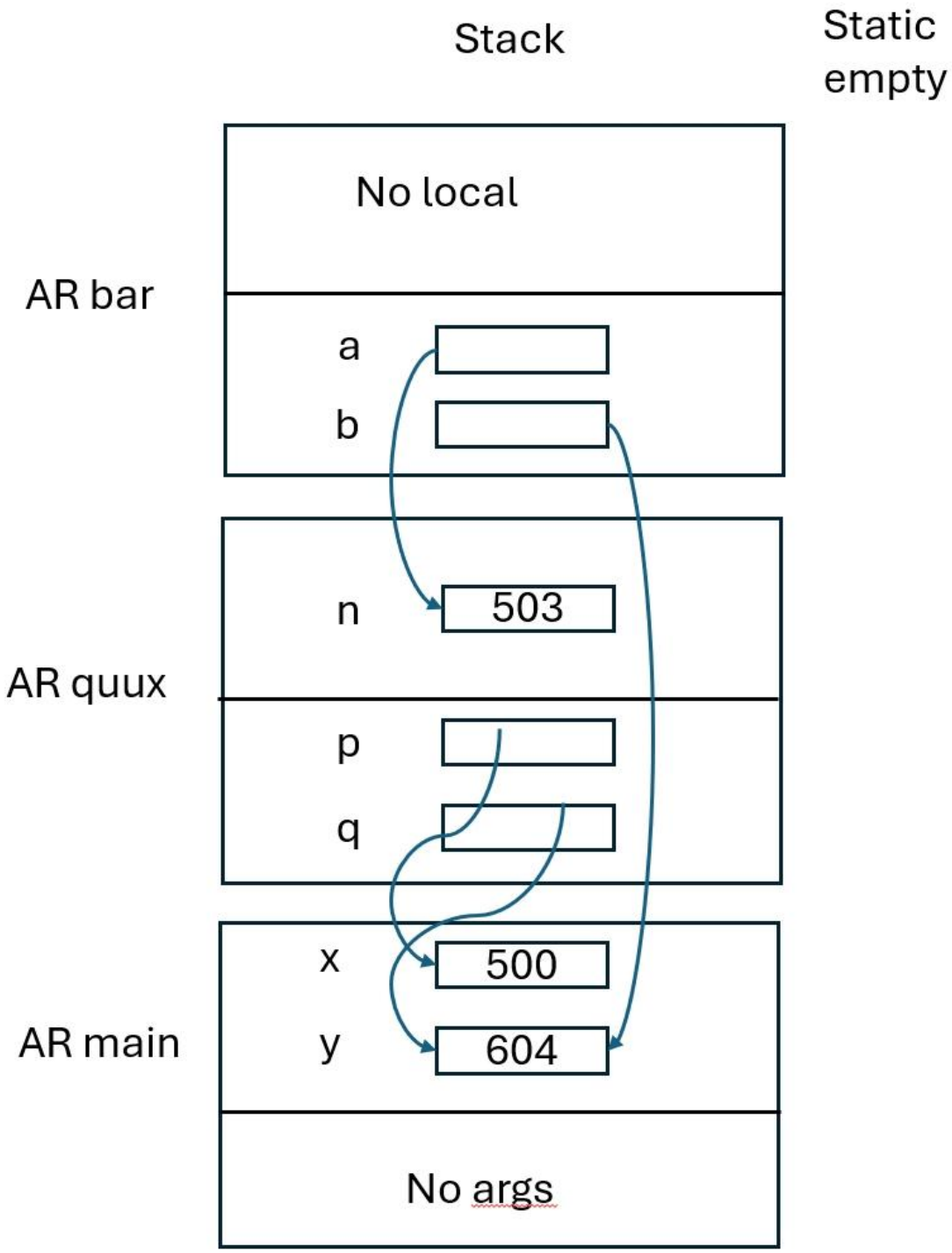
```
}

double Projectile_travel_time(double a, double v)

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

{
   double radian = degree_to_radian(a);
   return 2 * v * sin(radian) / G;
};

void create_table(double v)

{
   cout << "Angle\t\tt\td\n";
   cout << " (deg)\t\t(sec)\t\t(m)\n";
   for (int a = 0; a <= 90; a += 5)
   {
      double time = Projectile_travel_time(a, v);
      double distance = Projectile_travel_distance(a, v);
      cout << a << "\t\t" << time << "\t\t" << distance << "\n";
   }
}
```

Execution result

```
john2@John-Desktop /cygdrive/c/Users/john2/Desktop/uofc/c++/ENSF-604-assignment-repo
$ ./exercise_A.exe
Please enter the velocity at which the projectile is launched (m/sec): 33
Angle           t               d
 (deg)          (sec)           (m)
0               0               0
5               19.2962         0.586967
10              38.0061         1.16947
15              55.5612         1.74307
20              71.4281         2.3034
25              85.1247         2.8462
30              96.2349         3.36735
35              104.421         3.86286
40              109.434         4.32898
45              111.122         4.76215
50              109.434         5.15907
55              104.421         5.51674
60              96.2349         5.83242
65              85.1247         6.10371
70              71.4281         6.32854
75              55.5612         6.50521
80              38.0061         6.63238
85              19.2962         6.70907
90              -4.55832e-08            6.73469
```

Exercise B

Part II



Exercise C

```
/*
 *  lab1exe_C.cpp
 *  ENSF 694 Lab 1, exercise C
 *  Created by Mahmood Moussavi
 *  Completed by: John Zhou
 */

#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/60000;
  *seconds_ptr=ms_time%60000/1000.0;
}
```

Execution result

```
john2@John-Desktop /cygdrive/c/Users/john2/Desktop/uofc/c++/ENSF-604-assignment-repo
$  g++ -Wall lab1exe_C.cpp -o exercise_C

john2@John-Desktop /cygdrive/c/Users/john2/Desktop/uofc/c++/ENSF-604-assignment-repo
$ ./exercise_C.exe
Enter a time interval as an integer number of milliseconds: 3213
Doing conversion for input of 3213 milliseconds ...
That is equivalent to 0 minute(s) and 3.213 second(s).

john2@John-Desktop /cygdrive/c/Users/john2/Desktop/uofc/c++/ENSF-604-assignment-repo
$ ./exercise_C.exe
Enter a time interval as an integer number of milliseconds: 232323
Doing conversion for input of 232323 milliseconds ...
That is equivalent to 3 minute(s) and 52.323 second(s).

john2@John-Desktop /cygdrive/c/Users/john2/Desktop/uofc/c++/ENSF-604-assignment-repo
$
```

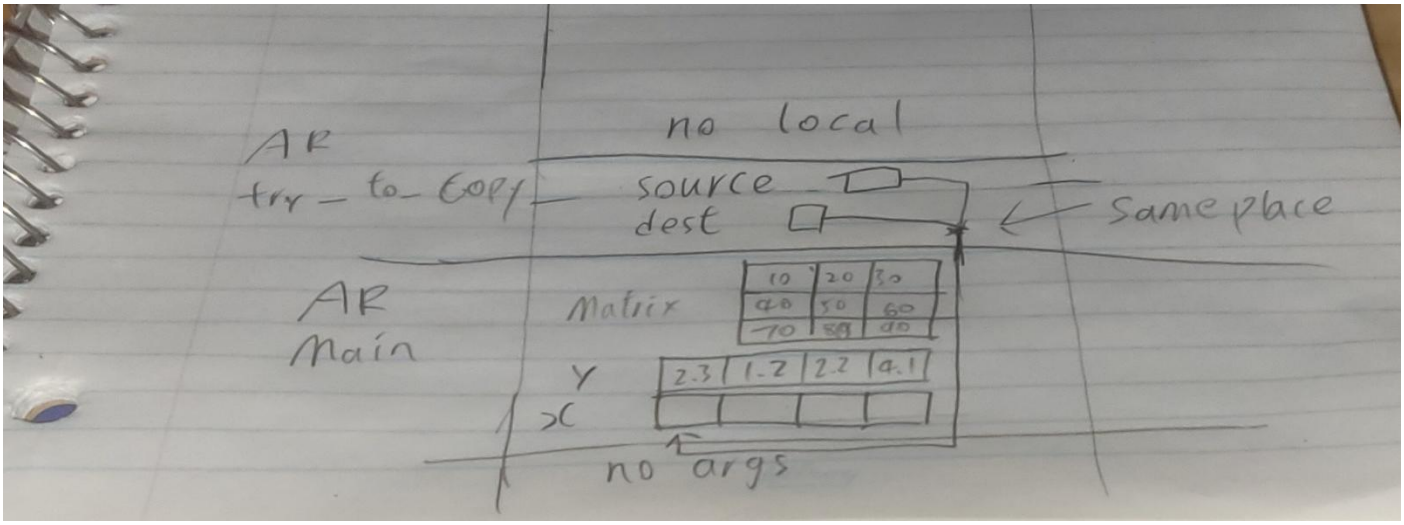
Exercise D

Part I

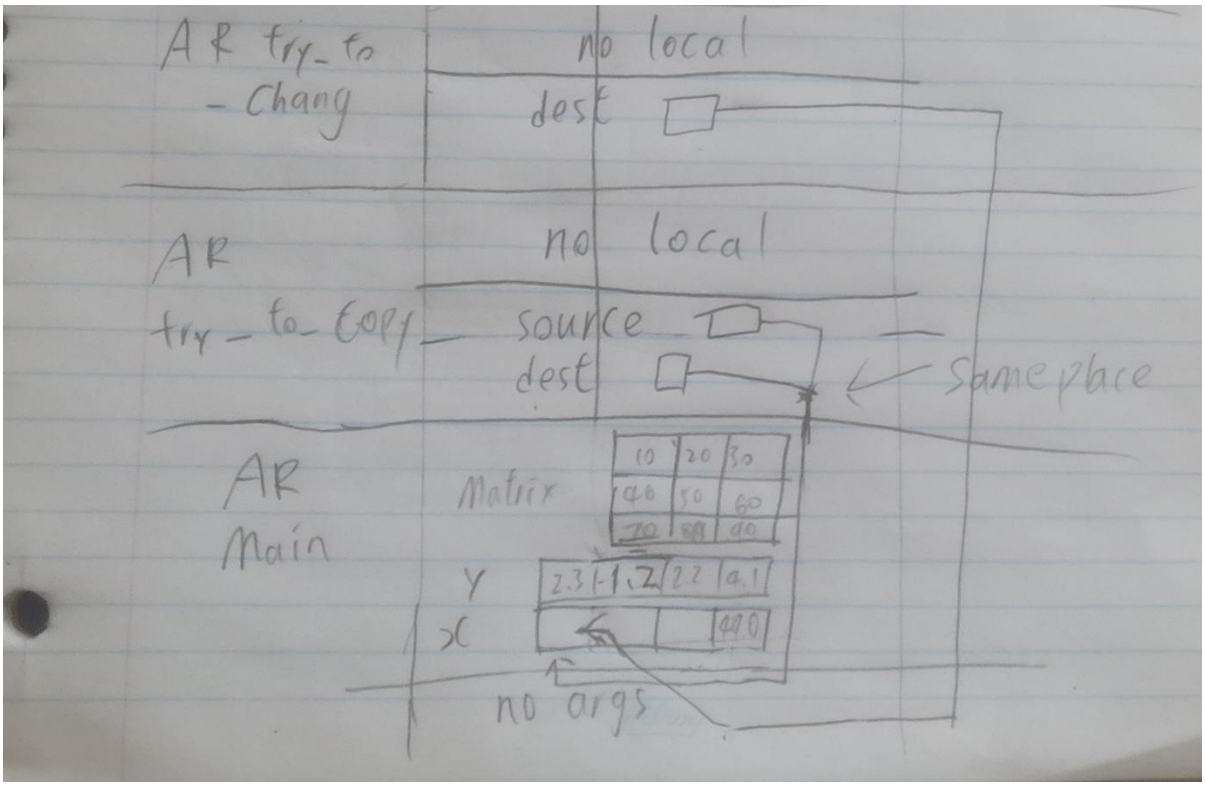
In the static area, there will be the Global constants
const int COL_SIZE = 3;
const int ROW_SIZE = 3;

The String constant area will have :
" sizeof(double) is \0", " bytes.\n \0" etc.

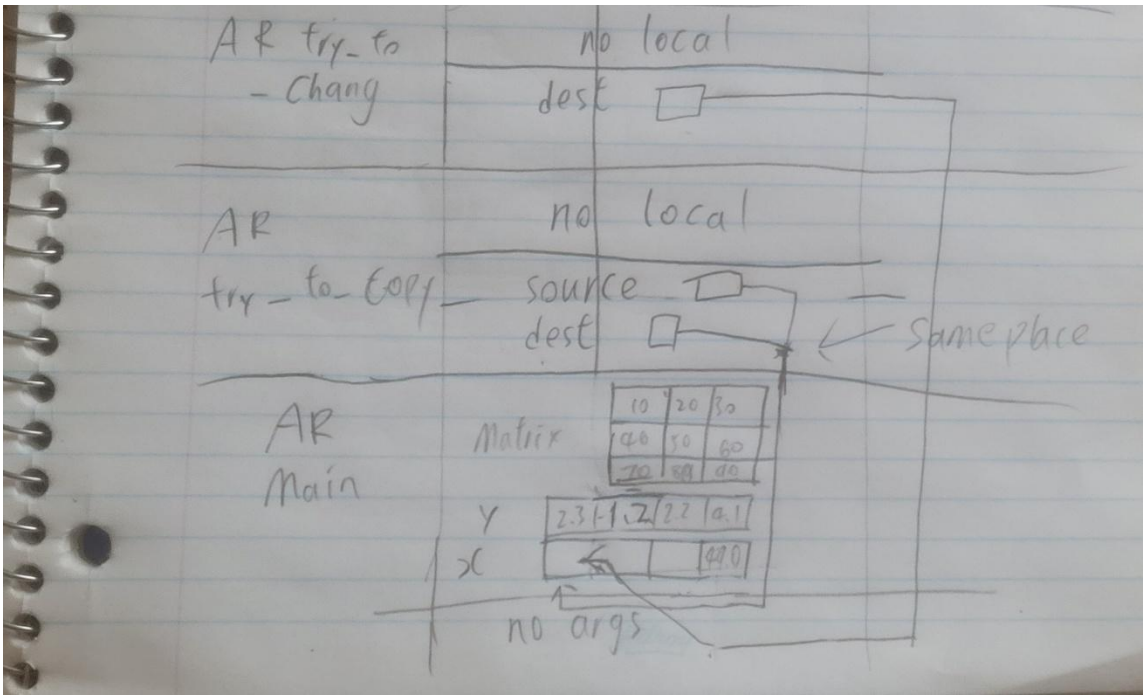
AR at Point 1 for stack



AR at Point 2 for stack



AR at Point 3 for stack



Part II

```
/*
 * lab1exe_D.cpp
 * ENSF 694 Lab 1, exercise D
 * Created by Mahmood Moussavi
 * Completed by: John Zhou
 */
#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)
{
```

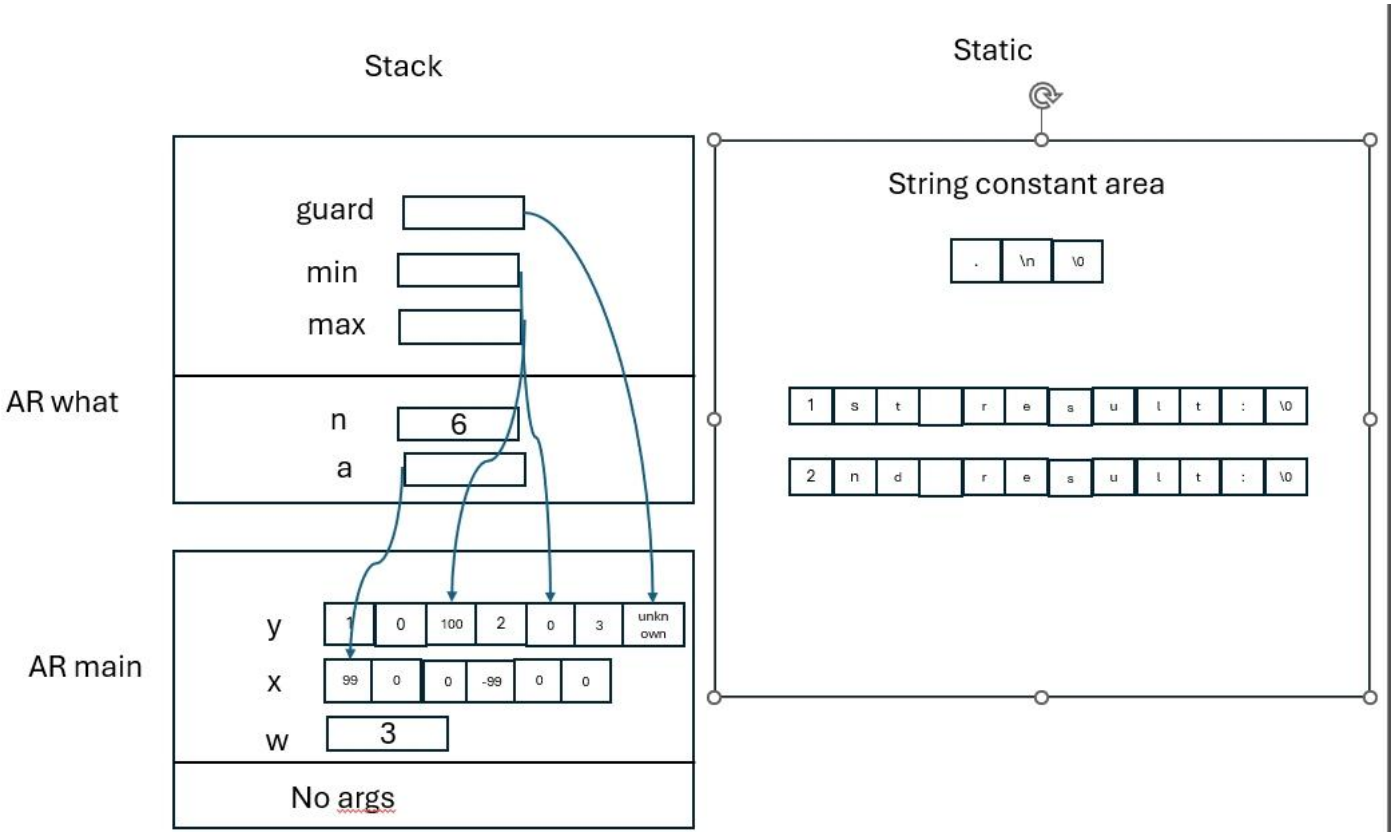
```
    dest[3] = 49.0;


    /* 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)
{
```

```
    cout << "_____Print Matrix_____\n";
    for (int i = 0; i <rows; i++)
    {
       for (int j = 0;j < COL_SIZE;j++){
          cout << matrix[i][j]<<" ";


       }
        cout << "\n";
    }
}
```

execution output

```
john2@John-Desktop /cygdrive/c/Users/john2/Desktop/uofc/c++/ENSF-604-assignment-repo
$ ./exercise_D.exe
 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.
 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:
_____Print Matrix_____
10.00 20.00 30.00
40.00 50.00 60.00
70.00 80.00 90.00

Program Ends...

john2@John-Desktop /cygdrive/c/Users/john2/Desktop/uofc/c++/ENSF-604-assignment-repo
$
```

Exercise E


Stack
Static
guard
min
max
AR what
n
6
a
y
1
0
100
2
0
3
unkn
own
AR main
x
99
0
0
-99
0
0
w
3
No args
String constant area
.
\n
\0
1 s t r e s u l t : \0
2 n d r e s u l t : \0

Exercise F

```
#include "MyArray.h"


int search(const MyArray* myArray, int obj)
{
    int currentSize = myArray->list_size;


   for (int i = 0; i < currentSize; i++)
   {
      if (myArray->array[i] == obj)
      {
         return i;
      }
   }


   return -1;
}
void initialize(MyArray *myArray)
{
   myArray->list_size = 0;
}


int retrieve_at(MyArray *myArray, int pos)
{


   int currentSize = size(myArray);
```

```
    if (pos >= 0 && pos < currentSize)
    {
        return myArray->array[pos];
    }

    return 0;
}

int count(MyArray *myArray, int obj)
{

    int occurances_count = 0;
    int currentSize = size(myArray);

    for (int i = 0; i < currentSize; i++)
    {
        if (myArray->array[i] == obj)
        {

            occurances_count++;
        }
    }

    return occurances_count;
}

void append(MyArray *myArray, int array[], int n)
{
    if (myArray->list_size + n <= SIZE)
```

```
    {
        int *start = myArray->array + myArray->list_size;
        for (int i = 0; i < n; i++)
        {
            start[i] = array[i];
        }
        myArray->list_size += n;
    }
}

void insert_at(MyArray *myArray, int pos, int val)
{
    int arrayCurrentSize = size(myArray);
    if (pos >= 0 && pos <= arrayCurrentSize)
    {

        for (int i = arrayCurrentSize; i > pos; i--)
        {
            myArray->array[i] = myArray->array[i - 1];
        }
        myArray->array[pos] = val;
        myArray->list_size++;
    }
}

int remove_at(MyArray *myArray, int pos)
{
    int arrayCurrentSize = size(myArray);
```

```
    if (pos >= 0 && pos < arrayCurrentSize)
    {
        int removedElement = myArray->array[pos];
        for (int i = pos; i < arrayCurrentSize-1; i++)
        {
            myArray->array[i] = myArray->array[i + 1];
        }

        myArray->list_size--;
        return removedElement;
    }
    return 0;
}

int remove_all(MyArray *myArray, int value)
{
    int countRemoved = 0;
    int currentSize = size(myArray);

    for (int i = 0; i < currentSize; i++)
    {
        if (myArray->array[i] == value)
        {
            remove_at(myArray, i);
            countRemoved++;
            i--;
            currentSize--;
        }
    }
```

```
    return countRemoved;
}

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

void display_all(MyArray *myArray)
{
    for (int i = 0; i < myArray->list_size; i++)
    {
        cout << myArray->array[i] << " ";
    }
    cout << endl;
}

bool is_full(MyArray *myArray)
{
    return myArray->list_size == SIZE;
}

bool isEmpty(MyArray *myArray)
{

    return myArray->list_size == 0;
}

int size(MyArray *myArray)
{
```

```
    return myArray->list_size;
}
```

Program output