**Course: ENSF 694** – Summer 2025

**Lab #:** 01

**Instructor:** Dr. Mahmood Moussavi

**Student Name:** Jack Shenfield

**Submission Date:** Wednesday, July 9th, 2025

**Exercise A**

Source code:

/\*

\* lab1exe\_A.cpp

\* ENSF 694 Lab 1, exercise A

\* Created by Mahmood Moussavi

\* Completed by: Jack Shenfield

\* Development Date: July 3rd, 2025

\*/

#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 << "Invalid input. Bye...";

exit(1);

}

}

create\_table(velocity);

return 0;

}

void create\_table(double v){

// Print titles

std::cout << "Angle\t" << "t\t" << "d" << std::endl;

std::cout << "(deg)\t" << "(sec)\t" << "(m)" << std::endl;

// Print 5 degree increments, up to 90 degrees

for(int i = 0; i <= (90/5); i++){

int thetadeg = i\*5; // angle in deg

double theta = degree\_to\_radian(i\*5); // angle in rad

double t = Projectile\_travel\_time(theta, v); // time

double d = Projectile\_travel\_distance(theta, v); // distance

std::cout << thetadeg << "\t" << t << "\t" << d << std::endl; // print statement

}

}

double Projectile\_travel\_time(double a, double v) {

return (2\*v\*sin(a))/G;

}

double Projectile\_travel\_distance(double a, double v) {

return ((pow(v, 2) / G) \* sin(2 \* a));

}

double degree\_to\_radian(double d) {

return(d \* (PI / 180));

}

Output (5 m/s inputted):

Please enter the velocity at which the projectile is launched (m/sec): 5

Angle t d

(deg) (sec) (m)

0 0 0

5 0.0889344 0.44298

10 0.177192 0.8725

15 0.264101 1.27551

20 0.349 1.63976

25 0.431243 1.9542

30 0.510204 2.20925

35 0.585282 2.39718

40 0.655906 2.51226

45 0.721538 2.55102

50 0.781678 2.51226

55 0.835869 2.39718

60 0.883699 2.20925

65 0.924804 1.9542

70 0.95887 1.63976

75 0.985639 1.27551

80 1.00491 0.8725

85 1.01653 0.44298

90 1.02041 -1.04645e-09

(base) jbs@Jacks-MacBook-Air ENSF694\_LabAssignment1 %

**Exercise BA diagram of a diagram

AI-generated content may be incorrect.**

**Exercise C**

Source code:

/\*

\* lab1exe\_C.cpp

\* ENSF 694 Lab 1 Exercise C

\* Completed by: Jack Shenfield

\* Development Date: July 3rd, 2025

\*/

#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", millisec;

/\* 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){

double seconds = (double)(ms\_time/1000); // seconds is the milliseconds divided by 100

int minutes = seconds/60; // number of minutes is the seconds/60 with no remainder

seconds = seconds - minutes\*60; // new seconds value, finds the remainder

\*minutes\_ptr = minutes; // call back to pointer

\*seconds\_ptr = seconds; // call back to pointer

}

Output:

Enter a time interval as an integer number of milliseconds: 150000

Doing conversion for input of 150000 milliseconds ...

That is equivalent to 2 minute(s) and 30 second(s).

(base) jbs@Jacks-MacBook-Air ENSF694\_LabAssignment1 %

**Exercise D**

Point 1:

**A paper with writing on it

AI-generated content may be incorrect.**

Point 2:

A diagram of a test

AI-generated content may be incorrect.

Point 3:

A diagram of a diagram with writing

AI-generated content may be incorrect.

Source code:

/\*

\* lab1exe\_D.cpp

\* ENSF 694 Lab 1 Exercise D

\* Completed by: Jack Shenfield

\* Development Date: July 5th, 2025

\*/

#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 each index in dest, copy the same value from the index in source

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

dest[i] = source[i];

}

// mising code -- students must complete the implementation of this funcion.

}

void print\_matrix(double matrix[][COL\_SIZE], int rows)

{ // for print statement runs for the # of rows

for(int i = 0; i < rows; i++){

cout << endl; // new line

for(int j = 0; j < COL\_SIZE; j++) // loop through each value in the row, tabbing over each time.

cout << matrix[i][j] << "\t";

}

// mising code -- students must complete the implementation of this funcion.

}

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...

(base) jbs@Jacks-MacBook-Air ENSF694\_LabAssignment1 %

**Exercise E**

Point 1:

A diagram of a diagram

AI-generated content may be incorrect.

**Exercise F**

Source code:

/\*

\* MyArray.cpp

\* ENSF 694 Lab 1 Exercise F

\* Completed by: Jack Shenfield

\* Development Date: July 5th, 2025

\*/

#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++) // search through the entire array, check if the obj values matches anything.

if(obj == myArray->array[i]){ // The first time it matches, return the index it matches at

return i;

}

return -1;

}

void initialize(MyArray\* myArray) {

// Students are supposed to complete the implementation of the this function

myArray->list\_size = 0; // set list size to 0

}

int retrieve\_at(MyArray\* myArray, int pos){

// Students are supposed to complete the implementation of the this function

return myArray->array[pos]; // retrieve the value from given position

}

int count(MyArray\* myArray, int obj ){

// Students are supposed to complete the implementation of the this function

int counter = 0; // initialize counter at 0

for(int i = 0; i < myArray->list\_size; i++){ // go through the entire array and increment counter when obj matches a value

if(obj == myArray->array[i]){

counter++;

}

}

return counter; // return the number of matches

}

void append( MyArray\* myArray, int array[], int n ) {

// Students are supposed to complete the implementation of the this function

if(myArray->list\_size + n <= SIZE) { // only continue this if there is space for n elements in the array

for(int i = myArray->list\_size; i < myArray->list\_size + n; i++){ // initialize i at current list length (the next empty index)

myArray->array[i] = array[i]; // store value

}

}

myArray->list\_size += n;

}

void insert\_at(MyArray\* myArray, int pos, int val) {

// Students are supposed to complete the implementation of the this function

for(int i = myArray->list\_size - 1; i >= pos; i--){ // starting at the end of the list, move all values over one

myArray->array[i + 1] = myArray->array[i];

}

myArray->array[pos] = val; // insert desired value

myArray->list\_size++; // increment list size

}

int remove\_at(MyArray\* myArray, int pos ) {

// Students are supposed to complete the implementation of the this function

int removed\_value = myArray->array[pos];

for(int i = pos; i < (myArray->list\_size - 1); i++){

myArray->array[i] = myArray->array[i + 1];

}

myArray->list\_size--;

return removed\_value;

}

int remove\_all(MyArray\* myArray, int value ) {

// Students are supposed to complete the implementation of the this function

int counter = 0; // initialize counter

for(int i = 0; i < myArray->list\_size; i++) { // if any array memory location == value, set it to 0

if(myArray->array[i] == value) {

myArray->array[i] = 0;

counter++; // increment counter when found

}

}

return counter; // return number of array elements removed

}

// 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

}

bool is\_full(MyArray\* myArray){

// Students are supposed to complete the implementation of the this function

if(myArray->list\_size == SIZE) {

return true; // if it is full return true

}

return false; // otherwise return false

}

bool isEmpty(MyArray\* myArray){

// Students are supposed to complete the implementation of the this function

if(myArray->list\_size == 0) {

return true; // if it is empty return true

}

return false; // otherwise return false

}

int size(MyArray\* myArray){

// Students are supposed to complete the implementation of the this function

return myArray->list\_size; // return the list size

}

Output:

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 ....