Course:Programming Fundamental – ENSF 337

Lab#: Lab 1

Instructor Name: Mahmood Moussavi

Student Name: Andy Wang and Sebastian Silva

Lab Section: L01

Date submitted: May 13, 2024

Exercise C. Operator Precedence (6 marks)

In the lectures, you were introduced to operator precedence in C. In this Task, you are asked to predict the value of the following expressions by using the proper operator precedence.
First, assume that:

```
double z = 0;
double x = 2.5;
double y = -1.5;
int m = 18;
int n = 4;
```

Now, what are the values of z in the following expressions? Show your work.

a) `z = x + n * y - (x + n) * y;`
b) `z = m / n + m % n;`
c) `z = n / m + n % m;`
d) `z = 5 * x - n / 5;`
e) `z = 1 - ( 1 - ( 1 - ( 1 - ( 1 - n))));`
f) `z = sqrt(sqrt((double)n);`

a) z = 2.5 + 4 * -1.5 - (2.5 + 4) * -1.5;
Z = 2.5 + -6.0 - (6.5) * -1.5;
Z = 2.5 + -6.0 + 9.75;
Z = 6.25;

b) Z = 18 / 4 + 18 % 4;
Z = 4 + 2;
Z = 6.0;

c) z = 4 / 18 + 4 % 18;
Z = 0 + 4;
Z = 4.0;

d) Z = 5 * x - n / 5;
Z = 5 * 2.5 - 4 / 5;
Z = 12.5 - 0;
Z = 12.5;

e) z = 1 - (1 - (1 - (1 - (1 - 4))));
Z = 1 - (1 - (1 - (1 + 3))));
z = 1 - (1 - (1 - 4));
z = 1 - (1 + 3);

z = 1 - 4;
z = -3.0;

f) Z = sqrt(sqrt((double)n));
Z = sqrt(sqrt(4.0));
Z = sqrt(2.0);
Z = 1.414213562;

Exercise D: Mathematical Expressions (5 marks)

For this task, write a program to read in an angle in units of radians and compute the sine of the angle. However, in addition to using the built-in sine function, you will also compute the Taylor series approximation, which is given by

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \cdots$$

The pseudo-code for your program is as follows:

- Prompt for input angle in units of radians.

- Read input angle in units of radians.
- Compute and output the sine of the input angle using built-in C math library function, called *sin*.
- Compute and output the sine of input angle using the **Taylor series approximation** **(only up to and including order seven: means,** x^7 **)**.
- If you are curious and add more terms, the Taylor series approximation gets closer to the results gained from using C math library, sin.

Write your source code in a file called **lab1_exe_D.c** and compile it into a program called **sine**.
Notes:

- Remember, in order to use the built-in $\sin(x)$ or $pow(x, y)$ functions, you must include the **math.h** file. To get you started you can use the following template.
- To calculate factorial of numbers, you can simply hardcode it. For example, for factorial of 7, use: 7*6*5*4*3*2

```
#include <stdio.h>
#include <math.h>

int main()
{
    double angle_in_radian;
    printf("Enter the angle in radian:\n");
    scanf("%lf", &angle_in_radian);
    printf("Sine of %f is %f.\n", angle_in_radian, sin(angle_in_radian));
    // Complete the program by calculating and displaying the Taylor's approximation.

   return 0;
}
```

Once you have written and compiled your code, test your program by running it and recording the outputs for following input values

a) 0 radians

b) 0.5 radians

c) 1.0 radians

d) 2.5 radians

```
#include <stdio.h>
#include <math.h>
int main()
{
	double angle_in_radian;
	printf("Enter the angle in radian:\n");
	scanf("%lf", &angle_in_radian);
	printf("Sine of %f is %f.\n", angle_in_radian, sin(angle_in_radian));
	// Complete the program by calculating and displaying the Taylor's approximation.

	printf("Sine of %f is %f.\n", angle_in_radian, (angle_in_radian -
pow(angle_in_radian,3.0)/6.0 + pow(angle_in_radian,5.0)/120.0 -
pow(angle_in_radian,7.0)/5040.0));

	return 0;
}
```

```
anste@DESKTOP-L5E6KDQ /cygdrive/c/Users/anste/ENSF337/lab1
$ ./sine.exe
Enter the angle in radian:
0
Sine of 0.000000 is 0.000000.
Sine of 0.000000 is 0.000000.

anste@DESKTOP-L5E6KDQ /cygdrive/c/Users/anste/ENSF337/lab1
$ ./sine.exe
Enter the angle in radian:
0.5
Sine of 0.500000 is 0.479426.
Sine of 0.500000 is 0.479426.

anste@DESKTOP-L5E6KDQ /cygdrive/c/Users/anste/ENSF337/lab1
$ ./sine.exe
Enter the angle in radian:
1
Sine of 1.000000 is 0.841471.
Sine of 1.000000 is 0.841468.

anste@DESKTOP-L5E6KDQ /cygdrive/c/Users/anste/ENSF337/lab1
$ ./sine.exe
Enter the angle in radian:
2.5
Sine of 2.500000 is 0.598472.
Sine of 2.500000 is 0.588534.
```

Radians	Sine Function	Taylor Series Approximation
0	0	0
0.5	0.479426	0.479426
1.0	0.841471	0.841468
2.5	0.598472	0.588534

Exercise F - Projectile Time and Distance Calculator

Read This First

In physics, assuming a flat Earth and no air resistance, a projectile launched with specific initial conditions will have a predictable range (maximum distance), and a predictable travel time.

The range or maximum horizontal distance traveled by the projectile can be approximately calculated by:

$$d = \frac{v^2}{g} \sin(2\Theta)$$

Where:

*g is gravitation acceleration (*9.81 m/s^2)
θ: the angle at which the projectile is launched in degrees
v: the velocity at which the projectile is launched
d: the total horizontal distance travelled by the projectile

To calulate the projectile travel time (t), when it reaches the maximum horizontal distace the followig formaula can be used :

$$t = \frac{2vsin\Theta}{g}$$

In this exercise you will complete a given C source file called `lab1exe_B.c` that prompts the user to enter a projectile's initial launch velocity (v) and displays the table of maximum horizontal distance and travel time for the trajectory angles of 0 to 90 degrees.

What to Do:

First, download file `lab1exe_F.c` from D2L. In this file the definition of function main and the function prototypes for four other functions are given. Your job is to complete the definition of the missing functions as follows:

Function create_table: which is called by the main function, receives the projectile initial velocity and displays a table of the projectile's maximum travel distance (d) and time (t), for trajectory angles of 0 to 90 (degrees), with increments of 5 degrees. Here is the sample of the required table:

```
 Angle          t           d
 (deg)        (sec)        (m)
0.000000     0.000000     0.000000
5.000000     1.778689     177.192018
10.000000    3.543840     349.000146
```


You don't have to worry about the format or the number of digits after the decimal point. The default format is acceptable.

Function projectile_travel_time: receives two double arguments, the trajectory angle (θ), and the initial velocity (v) and returns the projectile travel time (t).

Function projectile_travel_distance: receives two double arguments, the trajectory angle (θ), and the initial velocity (v) and returns the projectile maximum horizontal distance (d).

Function degree_to_radian: receives an angle in degrees and converts to radian. This function is needed, because the C library function `sin` needs its argument value to be in radian.

Notes:

- Please study the file posted on the D2L called: **`Interface Comments`**, and for each function in your program write an appropriate interface comment as instructed in this document.
- To use the C library function `sin`, you need to include header file `math.h`.
- When compiling from command line, if you are using Cygwin, Geany, or XCode, use `-lm` option to link the math library:

  ```
  gcc -Wall -lm lab1exe_F.c
  ```

- For constant values of π, and gravitation acceleration, `g`, the following lines are already included in the given file:

```
const double G = 9.8;
const double PI = 3.141592654;
```

```
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
```

```
/*
 *  Created by Mahmood Moussavi
 *  Completed by: Student Name
 */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

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)
{
   int n;
   double velocity;

   printf ("Please enter the velocity at which the projectile is launched (m/sec): ");
   n = scanf("%lf" ,&velocity);

   if(n != 1)
   {
      printf("Invlid input. Bye...");
      exit(1);
   }

   while (velocity < 0 )
   {
      printf ("please enter a positive number for velocity: ");
      n = scanf("%lf", &velocity);
      if(n != 1)
      {
         printf("Invlid input. Bye...");
         exit(1);
      }
   }

   create_table(velocity);
   return 0;
```

```
}

void create_table(double v){
	printf("Angle		t		d\n(deg)		(sec)		(m)\n");

	for(double angle = 0; angle <= 90; angle += 5){
		double radian = degree_to_radian(angle);
		printf("%f		%f		%f\n", angle, Projectile_travel_time(radian,
v), Projectile_travel_distance(radian, v));

	}
}

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

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;
}
/* UNCOMMENT THE CALL TO THE create_table IN THE main FUNCTION, AND COMPLETE
THE PROGRAM */
```

Exercise G: Drawing AR Diagrams for a Simple C Program

Read This First:

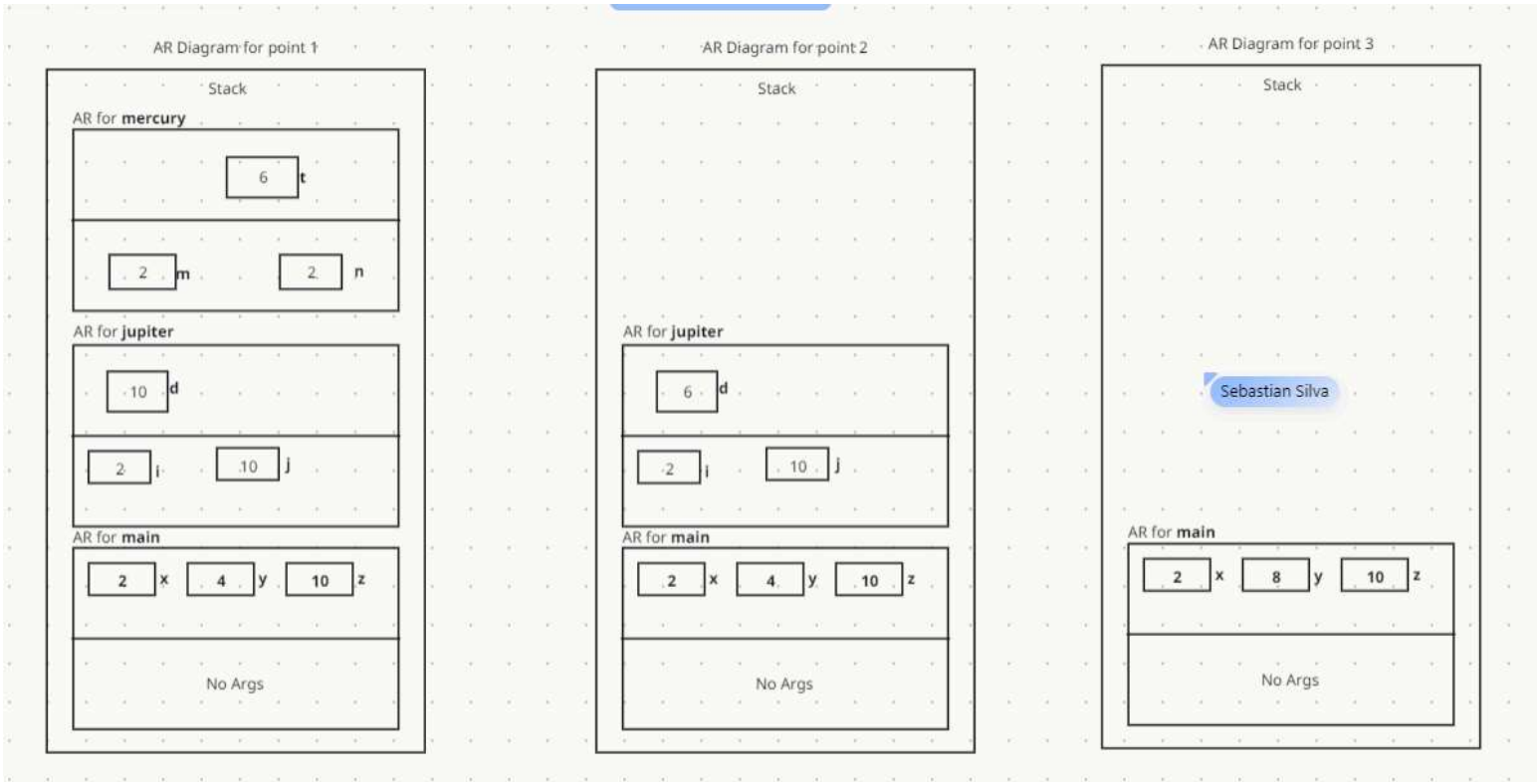
This is a very simple exercise on activation records. To learn more about activation records, please read the handout posted on the D2L, called: Activation Records:

What to Do

Download file `lab1exe_G.c` from D2L. Then, use pencil and paper to make memory diagrams for points one, two and three in the program.

What to Submit:

Submit your AR diagrams, as part of your lab report.


AR Diagram for point 1
Stack
AR for mercury
6
t
2
m
2
n
AR for jupiter
10
d
2
i
10
j
AR for main
2
x
4
y
10
z
No Args
AR Diagram for point 2
Stack
AR for jupiter
6
d
2
i
10
j
AR for main
2
x
4
y
10
z
No Args
AR Diagram for point 3
Stack
Sebastian Silva
AR for main
2
x
8
y
10
z
No Args

Exercise H – Introduction to Pointers

Read This First

This is an important exercise in ENSF 337. If you don't become comfortable with pointers, you will not be able to work with C. Spend as much time on this exercise as is necessary to understand exactly what is happening at every step in the given program.

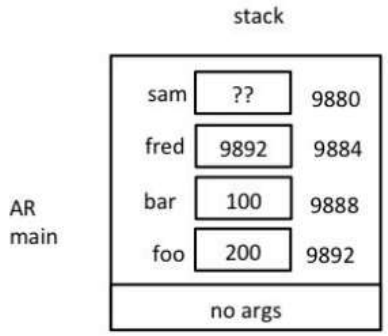
What to do

Download the file `lablexe_H.c`. Trace through the execution of the program to determine what the output will be.

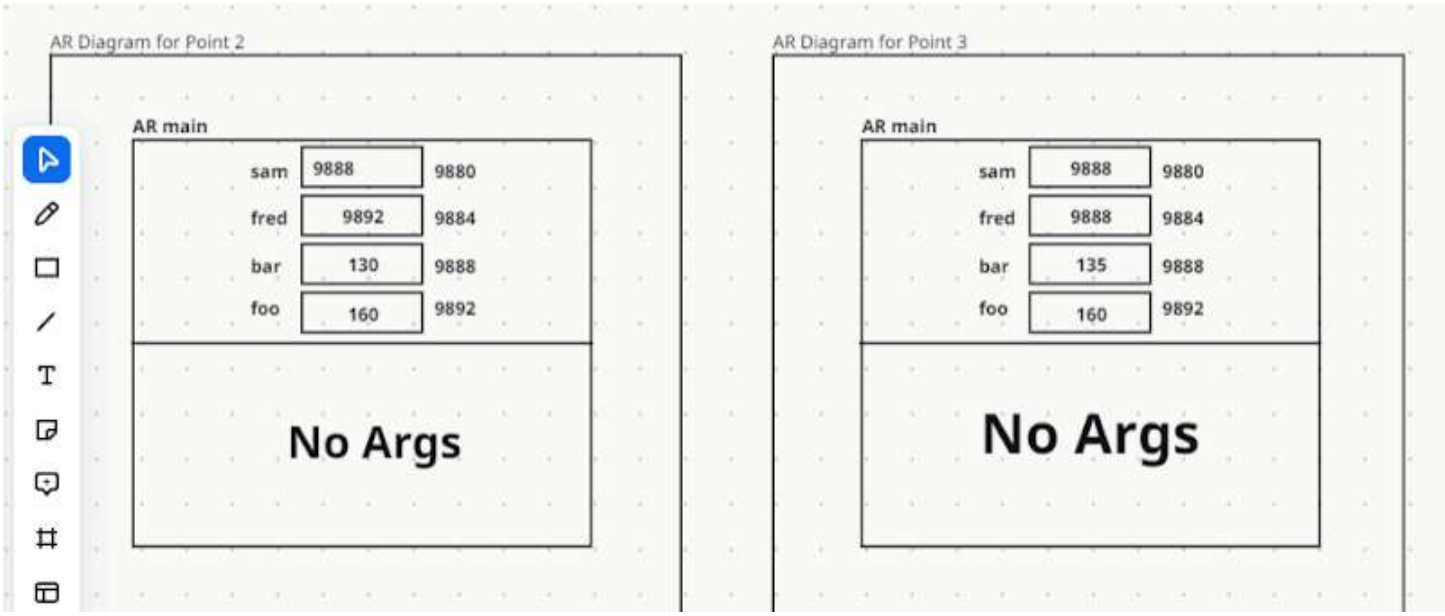
Now assume the following addresses for variables:

```
sam   9880
fred  9884
bar   9888
foo   9892
```

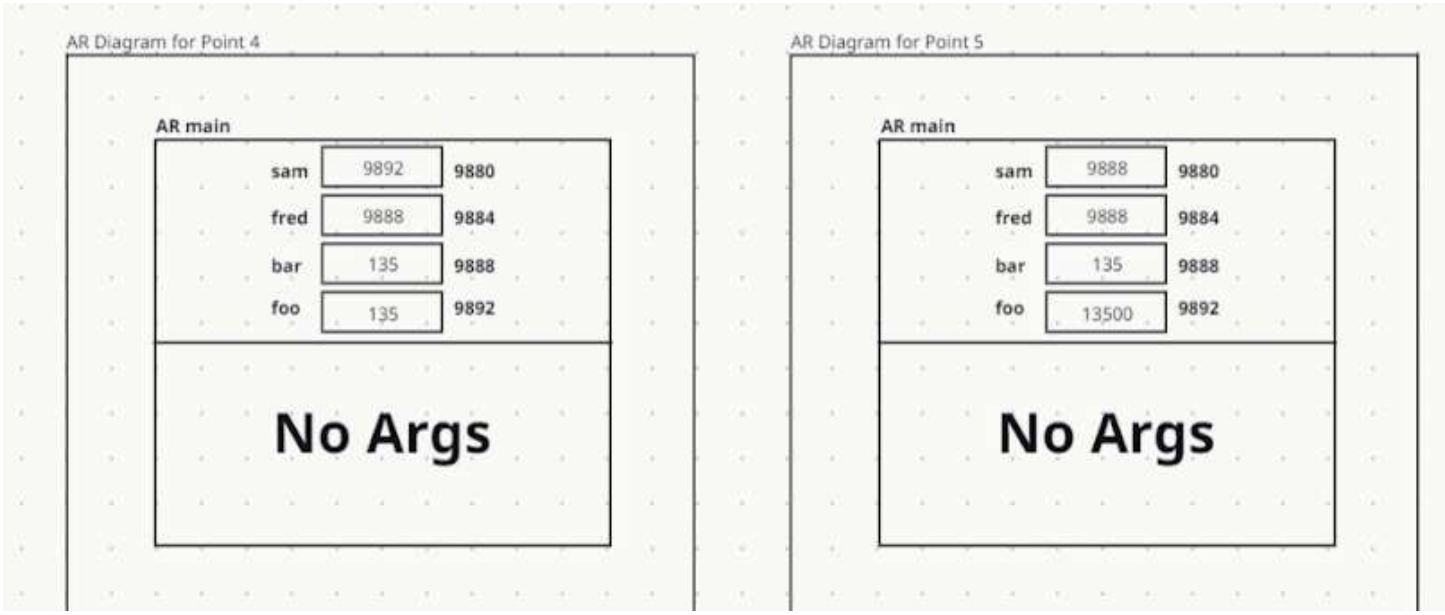
Draw a set of AR diagrams for points two through five and use the given address numbers as values of the pointers (don't use arrow notation in this exercise). To understand how to draw these diagrams the solution for point one is given in the following figure:



After you completed the exercise, compile lab2exe_C.c and check your predicted output against the actual program output.


AR Diagram for Point 2
AR main
sam 9888 9880
fred 9892 9884
bar 130 9888
foo 160 9892
No Args

AR Diagram for Point 3
AR main
sam 9888 9880
fred 9888 9884
bar 135 9888
foo 160 9892
No Args


AR Diagram for Point 4
AR main
sam 9892 9880
fred 9888 9884
bar 135 9888
foo 135 9892
No Args

AR Diagram for Point 5
AR main
sam 9888 9880
fred 9888 9884
bar 135 9888
foo 13500 9892
No Args

Exercise I: Pointers as Function Arguments

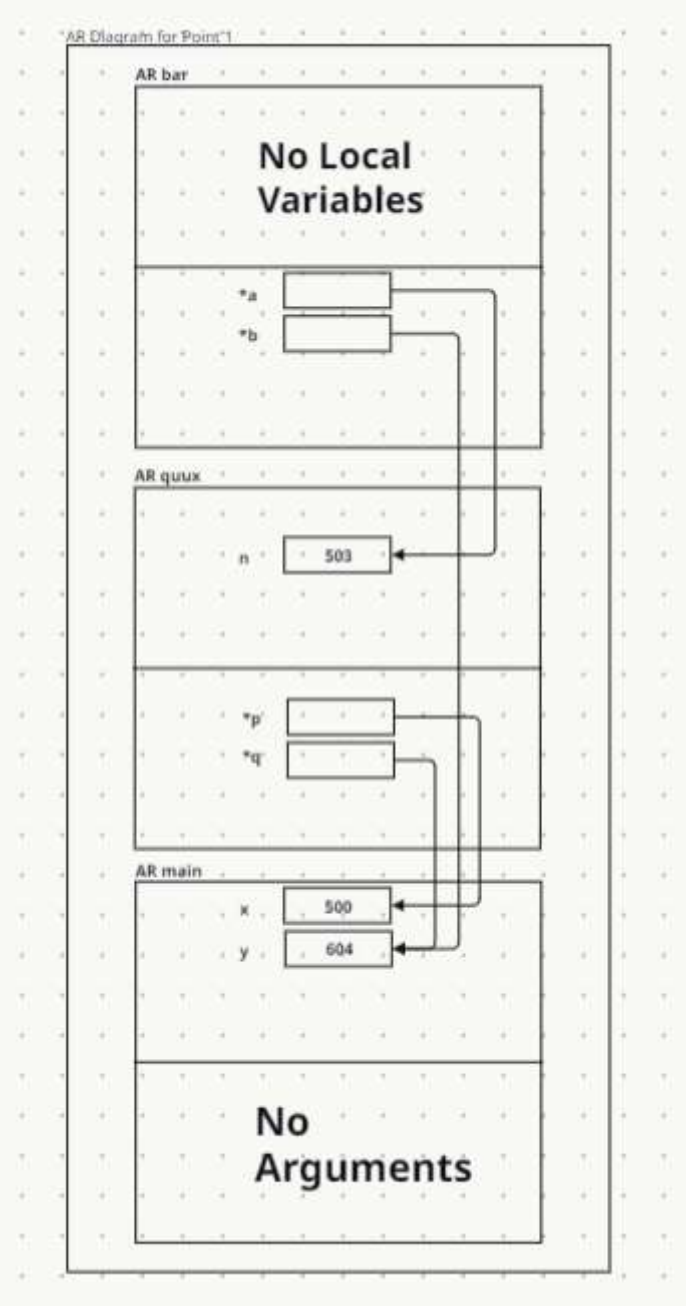
What to do – Part I

First copy the file `lab1exe_I1.c` from D2L. Carefully make diagrams for point one in this program using "**Arrow Notation**" as we discussed during lectures (you don't need to use made-up addresses). Then compare your solution with the posted solution on D2L.

There is nothing to submit for this part

What to do – Part II

Now download the file `lab1exe_I2.c` from D2L and draw AR diagram for point one in this file, for pointers using **arrow notation**.



Exercise J: Using pointers to get a function to change variables

What to do

Make a copy of the file `lab1exe_J.c` from D2L. If you try to compile and run this program it will give you some warning because the definition of function `time_convert` is missing.

Write the function definition for `time_convert` and add code to the main function to call `time_convert` before it prints answers.

```
#include <stdio.h>
#include <stdlib.h>

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;
  int nscan;

  do{
          printf("Enter a time interval as an integer number of milliseconds: ");
          nscan = scanf("%d", &millisec);
  }while(millisec < 0);

  if (nscan != 1) {
    printf("Unable to convert your input to an int.\n");
    exit(1);
  }


  printf("Doing conversion for input of %d ms ... \n", millisec);

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

  printf("That is equivalent to %d minute(s) and %f second(s).\n", minutes,
          seconds);

  return 0;
}
```

```
/* WRITE 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;

}
```