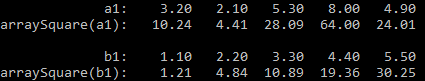
**Lab 3**

**Due Date: December 1, 2020, 11:59 PM**

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Q1 The goal of the program is to compute the position-wise squares of the values of an array. Complete that program, by defining an arraySquare function, that satisfies the following specs:

* Function arraySquare (...a...)
  + Has one input parameter called A, that is an array of double numbers
  + The function should return an array called result, with length equal to the length of A, such that the value at position i of result is square of the value at position i of A.
* Function printDoubleArray(...name[], ...a[], ...N)
  + has three arguments: **name** of the array, the array, and the length of the array
  + prints the name of the array followed by its contents
  + does not return anything.
* You are not allowed to change **main()** function in any way.
* The program generates the output shown below:



#include<stdio.h>

#define N 5

... printDoubleArray(... name..., ...a..., ...n) {

...............................

}

... arraySquare(... A...) {

.................................

}

int main() {

double a[] = {3.2,2.1,5.3,8.0,4.9};

double b[] = {1.1,2.2,3.3,4.4,5.50};

double \*result = arraySquare(a);

printDoubleArray("a", a, N);

printDoubleArray("arraySquare(a)", result, N);

result = arraySquare(b);

printf("\n");

printDoubleArray("b", b,N);

printDoubleArray("arraySquare(b)", result, N);

return 0;

}

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**Q2** The goal of the program is to merge the values of two multi-dimensional arrays into a single multidimensional array. Complete that program, by defining an arrayMerge function, that satisfies the following specs:

* Function arrayMerge (...result.., ...a...,...b...)
  + takes three arguments, called **result, a,** and **b**, that are arrays of char arrays.
  + merges the contents of **a** and **b** in the multidimensional array **result** that contains the values of **a**, in the order in which they appear in **a**, followed by the values of **b**, in the order in which they appear in **b**.
  + does not return anything
* Function printStringArray(...name..., ...a..., ... len)
  + has three arguments: **name** of the multidimensional array, the array, and the length of the array
  + prints the name of the array followed by its contents
  + does not return anything
* main() function
  + has three multidimensional arrays: result1, a1, and b1.
  + calls printStringArray() to print the contents of a1 and b1.
  + calls arrayMerge() to merge the contents of a1 and b1.
  + calls arrayMerge() to merge the contents of result1.
* You are not allowed to change **main()** function in any way.
* The program generates the output shown below:



#include <stdio.h>

#include <string.h>

....... max1 4

...... max2 8

...... strlen 10

void arrayMerge(.........){

.............................

}

.... printStringArray(.........)

{

.............................

}

int main() {

char a1[max1][ strlen] = {"Arlington", "Plano", "Laredo","Austin"};

char b1[max1][ strlen] = {"Amarillo", "McKinney", "Frisco", "Pasadena"};

char result1[max2][ strlen];

printf("\n");

printStringArray("a1", a1,max1);

printStringArray("b1", b1, max1);

arrayMerge(result1, a1, b1);

printStringArray("result1", result1,max2);

return 0;

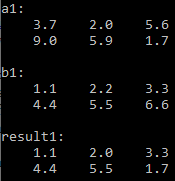
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Q3 The goal of the program is to compute the position-wise **minima** of two matrices (a matrix is a two-dimensional array). **Complete that program,** by defining a **matrixSmallest()** and **printDoubleMatrix()** functions, that satisfies the following specs

* Function **matrixSmallest**(...result..., ...A..., ....B...)
  + takes three arguments, called **result, A,** and **B**, that are **2D arrays** of double numbers
  + populates the value at position result[i][j] such that it is the smallest between the value at position A[i][j] and the value at position B[i][j].
  + does not return anything
* Function **printDoubleMatrix**(...name..., ...a..., ...rows1, ...cols1)
  + has four arguments: name of the multidimensional array, the array, rows in the array, cols in the array
  + prints the name of the array followed by its contents
  + does not return anything
* main() function
  + has three multidimensional arrays: result1, a1, and b1.
  + calls printDoubleMatrix () to print the contents of a1 and b1.
  + calls matrixSmallest() to compute the position-wise minima of two matrices
  + calls printDoubleMatrix () to print the contents of result1.
* You are not allowed to change **main()** function in any way.
* The program generates the output shown below:



#include <stdio.h>

#define rows 2

#define cols 3

void printDoubleMatrix(...name..., ...a..., ...rows1, ...cols1) {

......

}

void matrixSmallest(...result..., ...A..., ....B...) {

.......

}

int main() {

double a1[rows][cols] = {{3.7,2.0,5.6},{9.0,5.9,1.7}};

double b1[rows][cols] = {{1.1, 2.2,3.3},{4.4,5.5,6.6}};

double result1[rows][cols];

printDoubleMatrix("a1", a1, rows, cols);

printDoubleMatrix("b1", b1, rows, cols);

matrixSmallest(result1,a1, b1);

printDoubleMatrix("result1", result1, rows, cols);

return 0;

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Q4 The goal of the program is to compute the **average of values in specific regions of a matrix**. Complete that program, by defining a regionAverage() and **printDoubleMatrix()** functions, that satisfies the following specs:

* Function **regionAverage**(...A..., ...topRow, ...bottomRow, ...leftColumn, ...rightColumn) takes five arguments, called  **A, topRow**, **bottomRow**, **leftColumn**, and **rightColumn**. A is a 2D array of double numbers. Arguments top, bottom, left, right are all integers.
  + The function should return the average of values in all positions (i, j) of A such that topRow <= i <= bottomRow and leftColumn <= j <= rightColumn.
* Function **printDoubleMatrix**(...name..., ...a..., ...rows1, ...cols1)
  + has three arguments: name of the multidimensional array, the array, rows in the array, cols in the array
  + prints the name of the array followed by its contents
  + does not return anything
* main() function
  + has three multidimensional arrays: result1, a1, and b1.
  + calls printDoubleMatrix () to print the contents of a1 and b1.
  + calls regionAverage() twice with different arguments to compute the average of values in all positions (i, j) of A such that top <= i <= bottom and left <= j <= right.
  + print the average after each function call of regionAverage()
* You are not allowed to change **main()** function in any way.
* The program generates the output shown below:

**Example**: topRow=0, bottomRow=3, leftColumn=1, RightColumn=2

This means that use

rows 0-3 (highlighted blue)

columns 1-2 (colored red)

3.2 **2.1** **5.3**

8.0 **4.9** **5.7**

18.0 **14.9** **15.7**

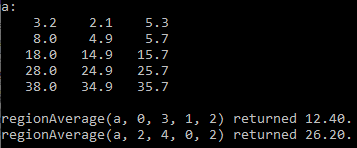
28.0 **24.9** **25.7**

38.0 **34.9** **35.7**

**Therefore, the average returned from regionAverage() will be (2.1+5.3+4.9+5.7+14.9+15.7+24.9+25.7) ÷ 8 => 12.4**

To calculate this region, we have chosen the area that is both highlighted blue and has red font.

In other words, this region is from row 0 to 3 and columns 1 to 2.



#include <stdio.h>

#define rows 5

#define cols 3

...**printDoubleMatrix**(...name..., ...a..., ...rows1, ...cols1) {

......

}

**...regionAverage(...A..., ...topRow, ...bottomRow, ...leftColumn, ...rightColumn)** {

......

}

int main() {

double a[rows][cols] = { {3.2,2.1,5.3}, {8.0,4.9,5.7}, {18.0,14.9,15.7}, {28.0,24.9,25.7}, {38.0,34.9,35.7} };

printDoubleMatrix("a", a, rows,cols);

double result = regionAverage(a, 0, 3, 1, 2);

printf("regionAverage(a, 0, 3, 1, 2) returned %.2f.\n", result);

result = regionAverage(a, 2, 4, 0, 2);

printf("regionAverage(a, 2, 4, 0, 2) returned %.2f.\n", result);

return 0;

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Q5 Write a program that

* Uses a 1 dimensional array:

arr1D[3] = {10, 20, 30};

* Uses a 2D array:

arr2D[2][3] = {{10, 20, 30},{40, 50, 60}};

* Uses a 3D array:

arr3D[4][2][3] =

{{{10, 20, 30},{40, 50, 60}},

{{11, 21, 31},{41, 51, 61}},

{{12, 22, 32},{42, 52, 62}},

{{13, 23, 33},{43, 53, 63}} };

* Completes the program given here.
* Prints an output as shown here:

#include<stdio.h>

int main() {

int arr1D[3] = {10, 20, 30};

printf("=====================================================================\n");

printf("1D Array\n");

...........................

printf("=====================================================================\n");

printf("2D Array\n");

int arr2D[2][3] = {{10, 20, 30},{40, 50, 60}};

...........................

printf("=====================================================================\n");

printf("3D Array\n");

int arr3D[4][2][3] = {{{10, 20, 30},{40, 50, 60}},

{{11, 21, 31},{41, 51, 61}},

{{12, 22, 32},{42, 52, 62}},

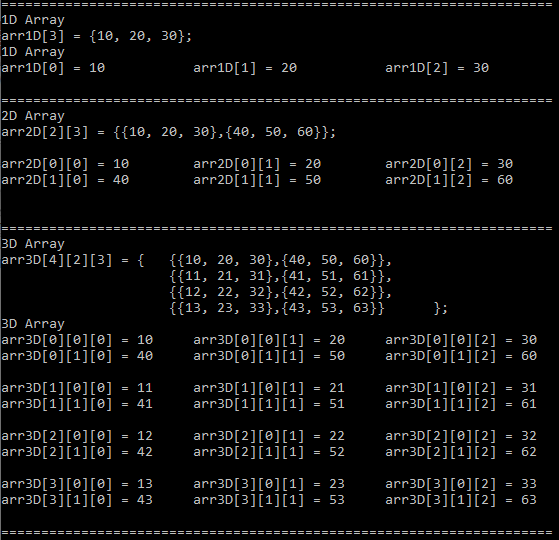
{{13, 23, 33},{43, 53, 63}} };

...........................

printf("=====================================================================\n");

...........................

}



////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Q6 Write a program that

* uses a 3D array “**arr**”

char **arr**.........= {

{{'A', 'B', 'C'}, {'D', 'E', 'F'}, {'G', 'H', 'I'} },

{{'J', 'K', 'L'}, {'M', 'N', 'O'}, {'P', 'Q', 'R'} },

{{'S', 'T', 'U'}, {'V', 'W', 'X'}, {'Y', 'Z', '-'} },

};

* Completes the following program
* Prints an output as shown here

#include<stdio.h>

**int main()**

{

char **arr**.....= {

{{'A', 'B', 'C'}, {'D', 'E', 'F'}, {'G', 'H', 'I'} },

{{'J', 'K', 'L'}, {'M', 'N', 'O'}, {'P', 'Q', 'R'} },

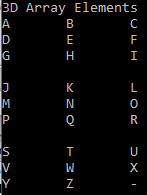
{{'S', 'T', 'U'}, {'V', 'W', 'X'}, {'Y', 'Z', '-'} },

};

for(i=0;i<3;i++){

........

}

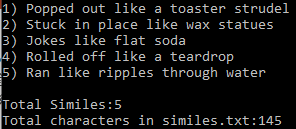


////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Q7 Write a program that:

* reads a file called similes.txt
* prints the message “**File not found**” if the file similes.txt is not found
* **counts** and **prints** the total number of **similes on the screen** and in a **file called output.txt**
* **counts** and **prints** the total number of **characters** in **similes.txt** **on the screen** and in a **file called output.txt**
* completes the program shown below and produces the given output



**Program output on screen and “output.txt” contents**

**Similies.txt (content of the file)**

Popped out like a toaster strudel

Stuck in place like wax statues

Jokes like flat soda

Rolled off like a teardrop

Ran like ripples through water

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <string.h>

int main()

{

............................

while (true){

..............................

}

..............................

}

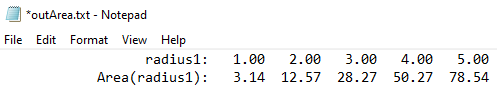
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

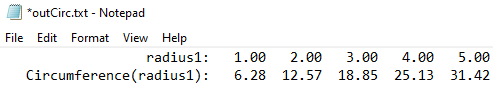
Q8 The goal of the program is to

* take **a one-dimensional array of radii** and
* create 2 **one-dimensional arrays** of corresponding **circumferences** and **areas**
* write the contents of array of radii and the array of circumferences in a file called **outCirc.txt**
* write the contents of array of radii and the array of areas(of circle) in a file called **outArea.txt**
* Complete the program, by defining
  + **printArray()** function, that satisfies the following specs:
    - Function printArray(....name...., ....a..., ....n)
    - Has **three input parameters** called **name, a, n**: Name of the array, array, and the number of elements in the array **a**
    - Prints the contents of array **a** on the screen
    - Does not return anything
  + **fprintArray()** function, that satisfies the following specs:
    - Has **four input parameters** called **f**, **name, a, n**: file identifier f, Name of the array, array, and the number of elements in the array **a**
    - Prints the contents of array **a** in the file associated with the file identifier **f**
    - Does not return anything
  + **arrayCirc**() function, that satisfies the following specs:
    - Has **one input parameter called A:** a one dimensional array of radii
    - Returns a pointer to an array of circumferences (of circle) corresponding to the array of radii A
  + **arrayArea**() function, that satisfies the following specs:
    - Has **one input parameter called A:** a one dimensional array of radii
    - Returns a pointer to an array of areas (of circle) corresponding to the array of radii A
* Produces an output as shown below
* Generates two files **outCirc.txt and outArea.txt** with contents as shown here
* Complete the main function

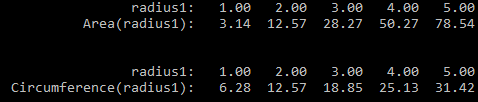
**Content of the generated file outArea.txt**



**Content of the generated file outCirc.txt**



**Screen Output**



#include<stdio.h>

#include<math.h>

#define N 5

**//prints the contents of an integer array**

**....printArray(....name...., ....a...., ....n)** {

....................

}

**//prints the contents of an integer array in a file identified by the file identifier f**

**....fprintArray(.... f, ....name...., ....a...., ....n)** {

...................

}

**//accepts an integer array (of radius) as an input parameter**

**//returns an array of Circumference**

**.... arrayCirc(.....A....)** {

.....................

}

**//accepts an integer array (of radius) as an input parameter**

**//returns an array of Area**

**..... arrayArea(....A....)** {

......................

}

**int main()** {

**...... outArea...="outArea.txt";**

**...... outCirc...="outCirc.txt";**

**........ fwArea=fopen(outArea,"w");**

**........ fwCirc=fopen(outCirc,"w");**

**// an array of radii**

**.......radius1... = {1.0,2.0,3.0,4.0,5.0,6.0};**

**//an array of circumferences**

**...................................................**

**//an array of areas**

**...................................................**

**//print the array of radius on screen**

**//print the array of radius in a file (outArea.txt)**

**//print the array of area on the screen**

**//print the array of area in a file (outArea.txt)**

**//print the array of radius on screen**

**//print the array of radius in a file (outCirc.txt)**

**//print the array of circumference on the screen**

**//print the array of circumference in a file (outCirc.txt)**

**// close files**

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Q9 The goal of the program is to read a file “**idioms.txt**” line by line. Then write the line in two files “**outFirst.txt**” and “**outlast.txt**”. After each line in “**outFirst.txt**” and “**outlast.txt**” print the first and the last word of the most recently written line respectively. Complete that program, by completing **main()** function and by defining functions **firstWord()** and **lastWord()** that satisfies the following specs:

* Function **firstWord(...line..., ... N);** 
  + has two input parameters, called line, a character array, and N, a variable of integer type
  + returns the address of the first word embedded in **line**.
* Function **lastWord(...line..., ... N);** 
  + has two input parameters, called line, a character array, and N, a variable of integer type
  + returns the address of the last word embedded in **line**.
* Function main()
  + reads from **idioms**.txt
  + writes each line from **idioms.txt** and its first word into **firstFirst.txt** as shown in the output
  + writes each line from **idioms.txt** and its last word into **lastFirst.txt** as shown in the output
  + if the file **idioms.txt** does not exist, displays the error message “**File not found**”
* The program generates the output shown below:

......................

......................

**... main()**

{ char bigline[200];

................................

**//if file does not exist, do error handling**

while (true){

{

................................................................

}

else ......... **//breaks out of loop when file has been read**

}

........................ **//don’t forget to close the file**

................................................................

}

**... firstWord(...line..., ...N)** {

................................................................;

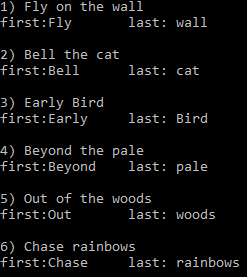
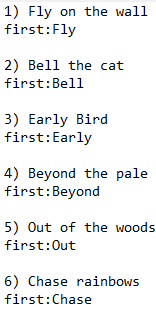
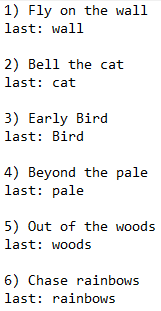
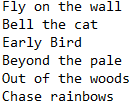
}

**... lastWord(...line..., ...N)** {

................................................................;

}

**Screen output outFirst.txt outlast.txt idioms.txt**

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**Q10 a** Convert the binary number **(**11010111**)₂**  to decimal representation. Show your work.

128 + 64 + 0 + 16 + 0 + 4 + 2 + 1 = 215

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**Q10 b** Convert the decimal number **(174)10**  to hexadecimal representation. Show your work.

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////