UNIVERSIDAD SERGIO ARBOLEDA
ESCUELA DE CIENCIAS EXACTAS E INGENIERÍA
Ciencias de la computación e Inteligencia Artificial
Arquitectura de Computadores
PARCIAL II

Estudiantes: Miguel Salazar, Diego Alejandro Bermúdez

Introducción:

Este taller-parcial tiene como propósito interiorizar los principios básicos de la programación en el lenguaje C, tales como manejo de variables, estructuras de control, manipulación de strings, manejo de archivos, implementación de algoritmos clásicos, entre otros. El criterio de selección de los ejercicios, por consiguiente, se basó en abarcar la mayor cantidad de elementos, de tal manera que a los autores permitiera sentirse cómodos con la programación en dicho lenguaje de programación.

Instrucciones de Instalación:

1. **Clonar el Repositorio parcial2Arquitectura (rama 'master') en un nuevo fichero**

   El repositorio está ubicado en el siguiente link:

   Repositorio (Click aquí)

   En caso de no funcionar, copiar y pegar el siguiente enlace en el navegador:

   https://github.com/miguelsalazar88/parcial2Arquitectura

2. En una pestaña de Terminal con ubicación en el fichero, ejecutar el Makefile mediante el comando $make$.

3. Elegir cualquiera de los ejecutables (por ejemplo sumaDeArray) y ejecutarlo en Terminal mediante el comando ./<*nombre del ejecutable*> . Por ejemplo: ./sumaDeArray

Ejercicios Implementados:

A continuación una breve explicación de cada ejercicio implementado, junto con su debida captura de pantalla de la ejecución:

Sección 1: “Simple C Questions”

1.1 Simple Interest

Ejercicio:	“Simple Interest”
Nombre Archivo:	**interesSimple.c**
Descripción:	Este programa calcula el interés simple de un préstamo. Se le pide al usuario el valor del préstamo, la tasa de interés (%) y el tiempo con base en la fórmula: $Valor\ total\ =\ Valor\ inicial\ + \frac{(Valor\ préstamo)\ *\ (Tasa\ de\ Interés)\ *\ (Tiempo)}{100}$
¿Requiere entrada de datos del usuario?	Sí.
Función Principal:	double interesSimple
Entrada:	unsigned long valor, int tasa, int tiempo
Salida:	double valorInteres

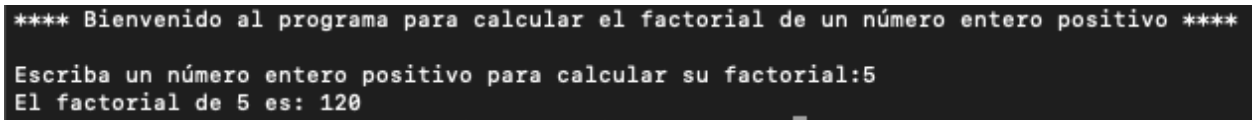
Captura de pantalla:

```
**** Bienvenido al programa que calcula el interés simple de un préstamo ****

Por favor ingrese el valor del préstamo en pesos: $100000
Por favor ingrese la tasa de interés en porcentaje: 12
Por favor ingrese el tiempo en años: 4
-----------------------------------------------
El valor del interés a pagar será de: $ 48000.000000
El gran total a pagar será de: $ 148000.000000
Miguels-MacBook-Pro:parcial2Arquitectura miguelsalazar$
```

1.2 Factorial of a Given Number

Ejercicio:	“Factorial of a Given Number”
Nombre Archivo:	**factorial.c**
Descripción:	Este es un programa que calcula el factorial de un número entero positivo ingresado por el usuario,mediante la utilización de una estructura de control for.
¿Requiere entrada de datos del usuario?	Sí.
Función Principal:	int factorial(int numero)
Entrada:	int numero
Salida:	int resultado

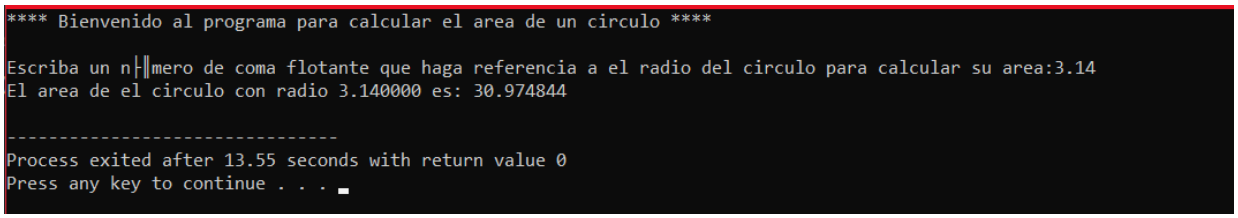
Captura de Pantalla:



1.3 Area of a Circle

Ejercicio:	“Area of a Circle”
Nombre Archivo:	**areaCirculo.c**
Descripción:	Este es un programa que calcula el área de un círculo partiendo de su radio.
¿Requiere entrada de datos del usuario?	Sí.
Función Principal:	float area(floar radio)
Entrada:	float radio
Salida:	float area

Captura de Pantalla:

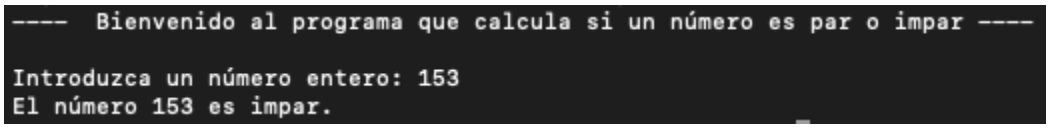


Sección 2: “If/Else Statement”

2.1 The Number is Even or Odd

Ejercicio:	“The Number is Even or Odd”
Nombre Archivo:	**parOImpar.c**
Descripción:	Este programa verifica si un número entero ingresado por el usuario es par o impar. Toma como base el funcionamiento del operador aritmético del residuo (%), dado que el residuo de un número par dividido dos es siempre igual a cero.
¿Requiere entrada de datos del usuario?	Sí.
Función Principal:	esPar(int num)
Entrada:	int num
Salida:	bool esPar

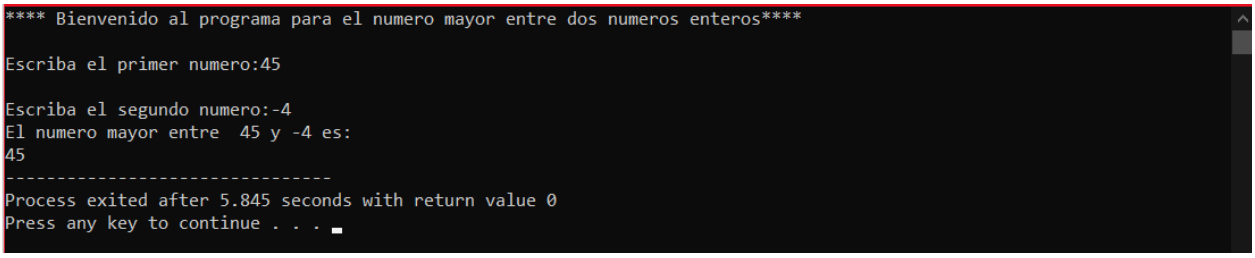
Captura de Pantalla:



2.2 Find The Maximum Between Two Numbers

Ejercicio:	“Find The Maximum Between Two Numbers”
Nombre Archivo:	**numeroMayor.c**
Descripción:	Este programa verifica cuál número (de dos ingresados por el usuario) es mayor.
¿Requiere entrada de datos del usuario?	si
Función Principal:	void max(int a, intb)
Entrada:	int a, int b
Salida:	void (La función no retorna ningún valor pero se muestra el número que sea mayor).

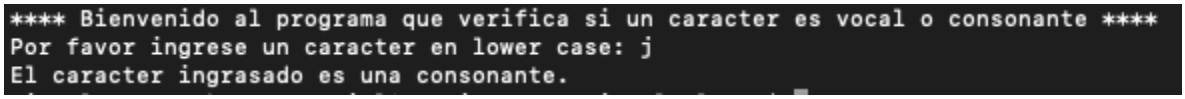
Captura de Pantalla:



2.3 Input Any Alphabet and Check Whether it is Vowel or Consonant

Ejercicio:	“Input any Alphabet and Check Whether it is Vowel or Consonant”
Nombre Archivo:	**vocalOconsonante.c**
Descripción:	Este programa verifica si un caracter ingresado por el usuario es vocal o consonante. Se utiliza la estructura de control if que verifica si el caracter corresponde a (‘a’ || ’e’ || ’i’ || ’o’ || ’u’).
¿Requiere entrada de datos del usuario?	Sí.
Función Principal:	void vocalOConsonante(char c)
Entrada:	char c
Salida:	void

Captura de Pantalla:

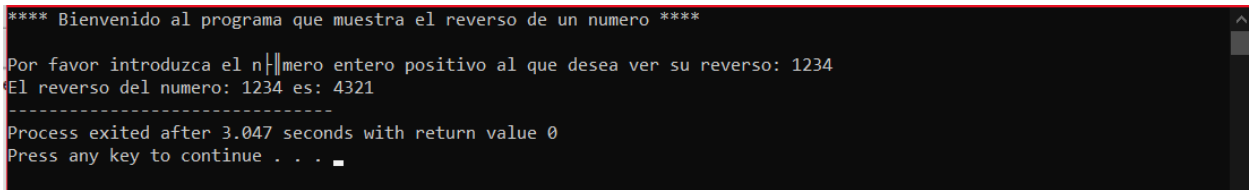


Sección 3: “Loops”

3.1 Reverse a Given Number

Ejercicio:	“Reverse a Given Number”
Nombre Archivo:	**reversoDeUnNumero.c**
Descripción:	Este programa muestra el reverso de un número ingresado por el usuario.
¿Requiere entrada de datos del usuario?	si
Función Principal:	int rev(int numero)
Entrada:	int numero
Salida:	int reverso

Captura de Pantalla:



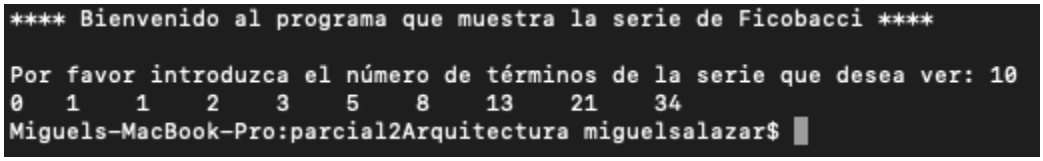
3.2 Display Fibonacci Series

Ejercicio:	“Display Fibonacci Series”
Nombre Archivo:	**fibonacci.c**

Descripción:	Este programa muestra la cantidad de términos de la serie de Fibonacci introducida por el usuario. Utiliza la estructura de control while de la siguiente manera, expuesta en pseudocódigo:
	
	num1 = 0
	num2 = 1
	i
	
	While (i < inputUsuario) do
	  suma = num1+num2
	  num2 = suma
	  num1 = num2
	end

¿Requiere entrada de datos del usuario?	Sí.
Función Principal:	No se utilizan funciones en este ejercicio
Entrada:	int n
Salida:	int suma

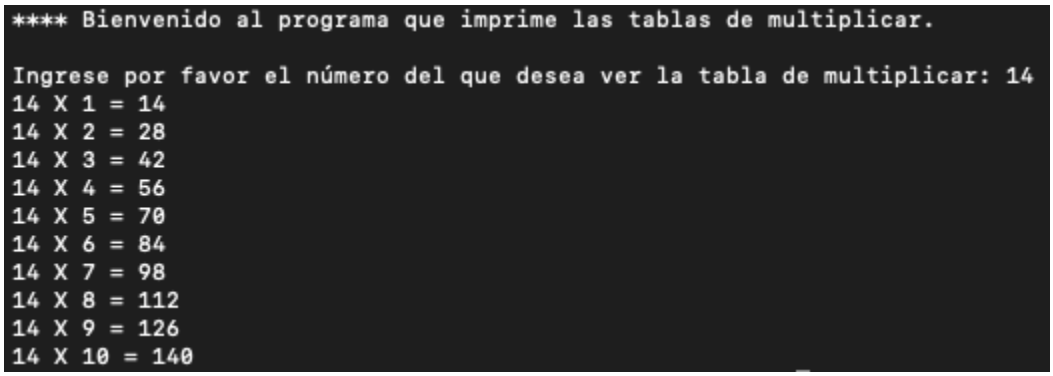
Captura de Pantalla:



3.3 Print Multiplication Table Using (For Loop)

Ejercicio:	“Print Multiplication Table Using (For Loop)”
Nombre Archivo:	**tablaMultiplicacion.c**
Descripción:	Este programa, mediante la implementación de una estructura de control tipo for, itera entre los números 1 y 10, y los multiplica por el número ingresado por el usuario para de esta manera imprimir su tabla de multiplicación.
¿Requiere entrada de datos del usuario?	Sí.
Función Principal:	void tablaMultiplicacion(int n)
Entrada:	int n
Salida:	Void (imprime $n \times i$ en cada iteración)

Captura de Pantalla:

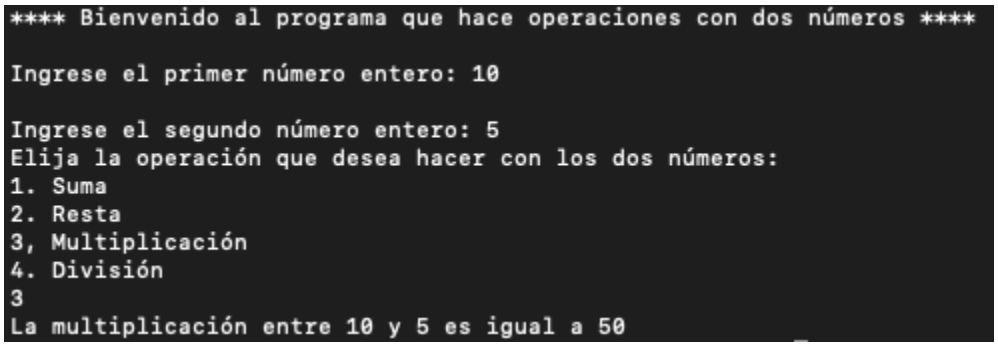


Sección 4: Switch Case

4.1 Calculator

Ejercicio:	“Calculator”
Nombre Archivo:	**calculadora.c**
Descripción:	Este programa, mediante la implementación de la estructura condicional Switch, realiza operaciones entre dos números ingresados por el usuario.
¿Requiere entrada de datos del usuario?	Sí.
Funciones:	int suma(int uno, int dos) int resta(int uno, int dos) int multiplicacion(int uno, int dos) double división(int uno, int dos)
Entrada:	int uno, int dos
Salida:	int suma, int resta, int multiplicacion, double division

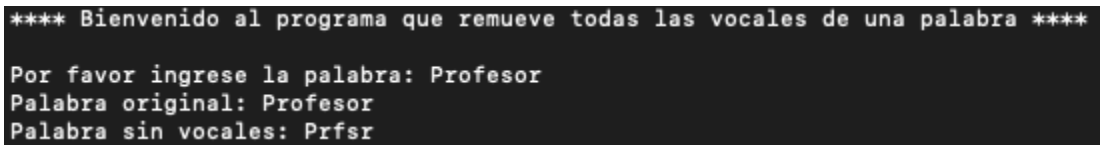
Captura de Pantalla:



4.2 Remove All Vowels From a String

Ejercicio:	“Remove all Vowels Form a String”
Nombre Archivo:	**removerVocales.c**
Descripción:	Este programa, mediante la implementación de una estructura condicional Switch, reemplaza todos aquellos caracteres de un string (introducido por el usuario) que correspondan a una vocal por un caracter vacío (‘\0’). Posteriormente, se imprime el arreglo omitiendo los caracteres vacíos.
¿Requiere entrada de datos del usuario?	Sí.
Función Principal:	void removerVocales(char string[ ])
Entrada:	char string[ ]
Salida:	void

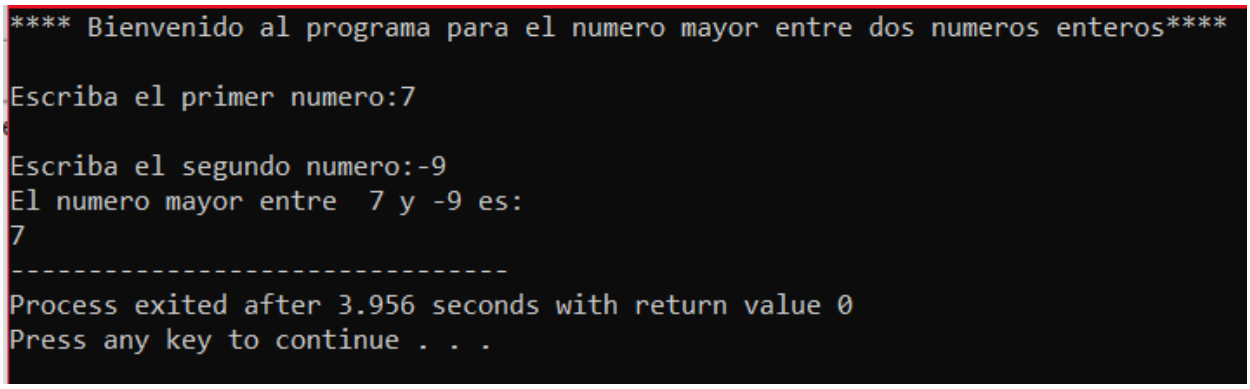
Captura de Pantalla:



4.3 Find The Maximum Between Two Numbers Using Switch Case

Ejercicio:	"Find the Maximum Between Two Numbers Using Switch Case"
Nombre Archivo:	**numeroMayorSwitchCase.c**
Descripción:	Este programa determina cual es el número mayor (de dos ingresados) empleando Switch Case
¿Requiere entrada de datos del usuario?	si
Funciones:	void max(int a,int b)
Entrada:	int a,b
Salida:	void "muestra que numero es mayor o si son iguales"

Captura de Pantalla:

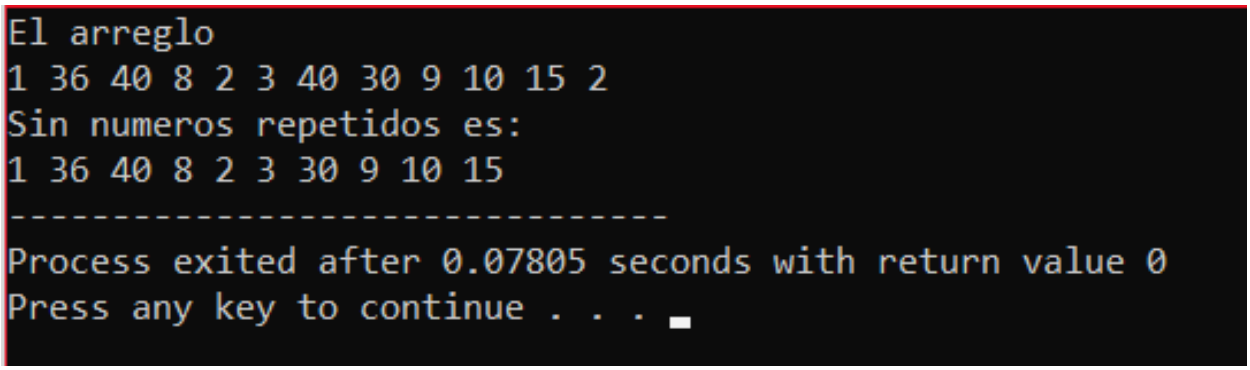


Sección 5: “Array Questions”

5.1 Remove Duplicate Items in An Array

Ejercicio:	“Remove Duplicate Items In An Array”
Nombre Archivo:	**removerRepetidosArreglo.c**
Descripción:	Este programa elimina los números repetidos de un arreglo predeterminado y muestra el arreglo resultante
¿Requiere entrada de datos del usuario?	no
Funciones:	void remover(int numero[],int n[],int tamanio) int contiene(int n,int numeros[],int tamanio)
Entrada:	“Ingresa un arreglo predeterminado”
Salida:	“Sale el arreglo sin números repetidos”

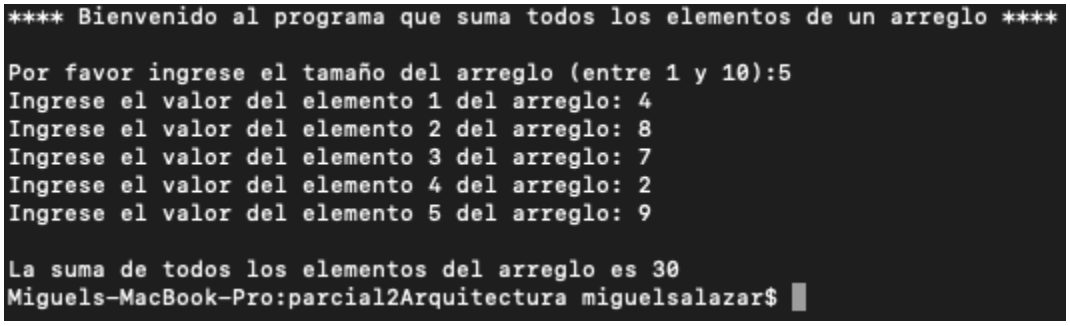
Captura de Pantalla:



5.2 Sum of All Array Elements Using Recursion

Ejercicio:	“Sum of All Array Elemants Using Recursion”
Nombre Archivo:	**sumaDeArray.c**
Descripción:	En este programa se utiliza una función recursiva para sumar todos los elementos de un arreglo cuyos valores son introducidos por el usuario. Toma como caso base que el índice (int inicio) sea igual al tamaño del arreglo. De no ser así, procede a llamar nuevamente a la función.
¿Requiere entrada de datos del usuario?	Sí.
Función Principal:	int sumaDeArray(int arr[ ], int inicio, int tamanio)
Entrada:	int arr[ ], int inicio, int tamanio
Salida:	(arr[inicio] + sumaDeArray(arr, inicio+1, tamanio)) recursivamente hasta llegar al caso base.

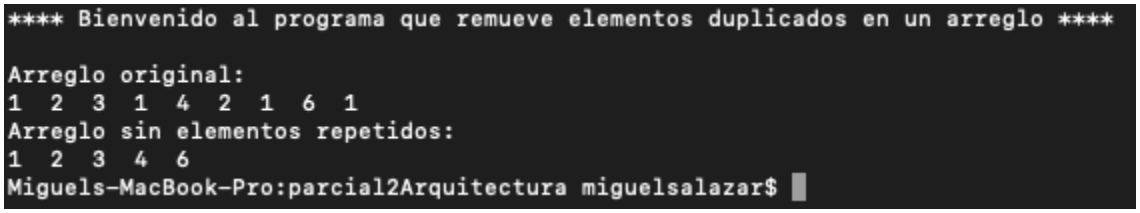
Captura de Pantalla:



5.3 Delete All Duplcate Elements From An Array

Ejercicio:	“Delete All Duplicate Elements From An Array”
Nombre Archivo:	**removerDuplicados.c**
Descripción:	En este programa se utilizan tres estructuras if anidadas para lograr remover los elementos duplicados: El primero tiene como propósito recorrer el arreglo, el segundo es utilizado para comparar los elementos i yj, y el tercero tiene el propósito desplazar hacia la izquierda todos los elementos que suceden el elemento duplicado.
Función Principal:	void removerDuplicados(int arr[ ], int tamanio)
Entrada:	int arr[ ], int tamanio
Salida:	void

Captura de Pantalla:

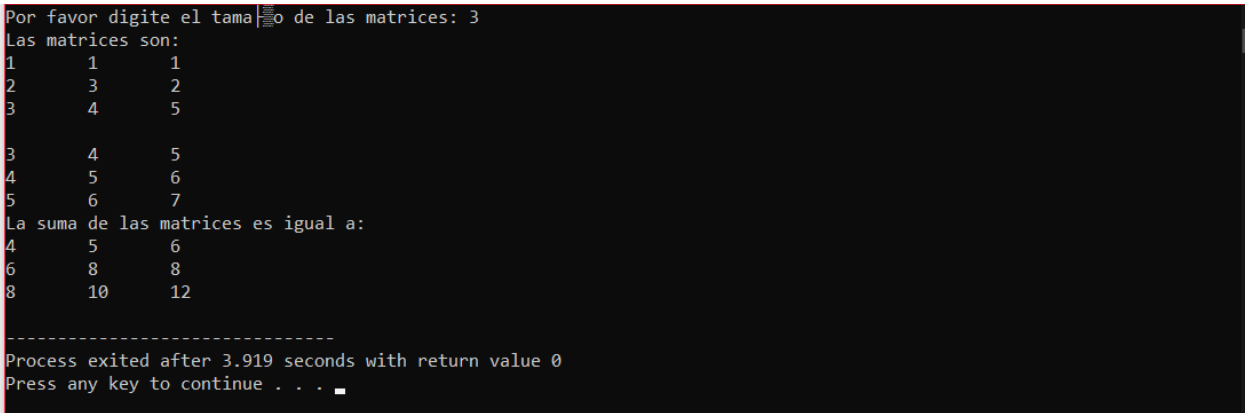


Sección 6: Matrix Questions

6.1 Add Two Matrices

Ejercicio:	“Add Two Matrices”
Nombre Archivo:	**sumaMatrices.c**
Descripción:	Este programa suma dos matrices cuadradas de tamaño ingresado por el usuario generadas por la máquina y muestra su resultado
¿Requiere entrada de datos del usuario?	si
Funciones:	void initMatriz(int n, int matriz[n][n],int y) void imprimirMatriz(int n, int matriz[n][n]) void sumaMatrices(int n, int matriz[n][n],int matriz2[n][n])
Entrada:	int tamanio
Salida:	int matriz[tamanio][tamanio]

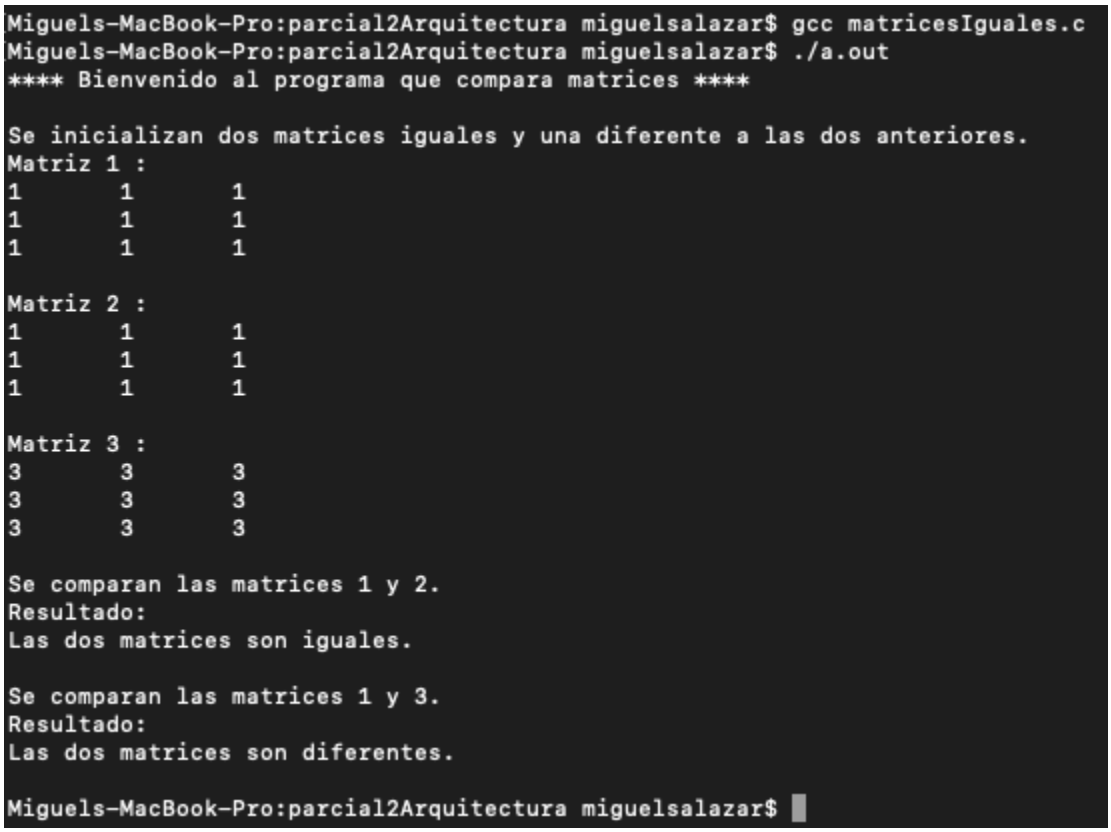
Captura de Pantalla:



6.2 Two Matrices Are Equal or Not

Ejercicio:	“Two Matrices Are Equal or Not”
Nombre Archivo:	**matricesIguales.c**
Descripción:	En este programa, mediante la utilización de dos estructuras for anidadas, compara tres matrices A, B y C, de las cuales A = B y A !=C, con el fin de demostrar el correcto funcionamiento del mismo.
¿Requiere entrada de datos del usuario?	No.
Función Principal:	void compararMatrices(int n, int matrizA[n][n], int matrizB[n][n])
Entrada:	int n, int matrizA[n][n], int matrizB[n][n]
Salida:	void

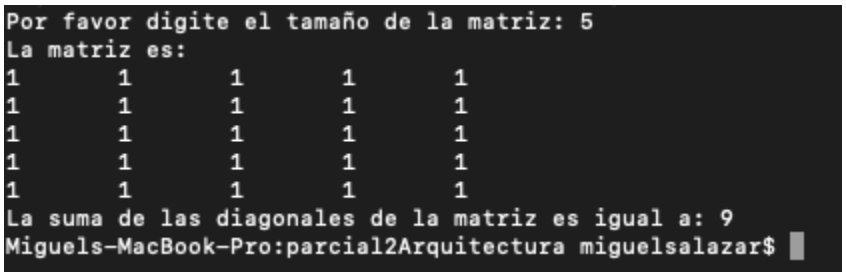
Captura de Pantalla:



6.3 Sum of the Diagonals of A Matrix

Ejercicio:	“Sum of the Diagonals of A Matrix”
Nombre Archivo:	**sumaDiagonalMatrices.c**
Descripción:	Este programa, mediante la implementación de dos estructuras for, suma los elementos que componen las dos diagonales de una matriz cuadrada. Toma como base el hecho de que todos los elementos de la diagonal 1 cumplen con la regla de que el número de índice de la fila es igual a aquel de la columna ($i = j$) y, asimismo, todos los elementos de la diagonal 2 cumplen con la regla de que la suma de los índices de fila y columna es igual al tamaño de la matriz.
¿Requiere entrada de datos del usuario?	Si.
Función Principal:	int sumaDiagonales(int n,int matriz[n][n]);
Entrada:	int n,int matriz[n][n]
Salida:	int suma

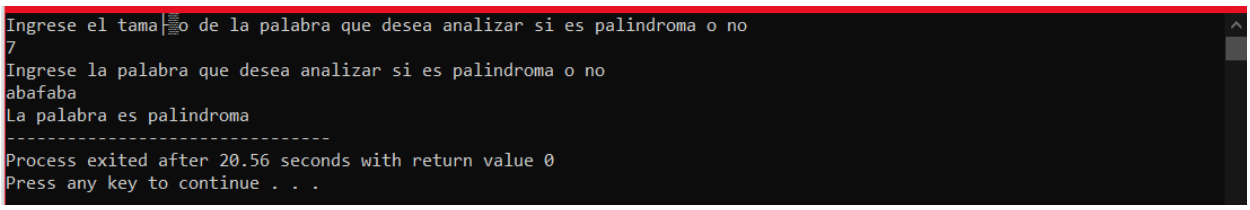
Captura de Pantalla:



Sección 7: “String Questions List”
7.1 A String is Palindrome or Not

Ejercicio:	“A String is Palindrome or Not”
Nombre Archivo:	**palindroma.c**
Descripción:	Este programa lee una palabra, de tamaño elegido, ingresada por el usuario y muestra si es o no palindroma
¿Requiere entrada de datos del usuario?	si
Función Principal:	void palindroma(int tamanio,char palabra[])
Entrada:	int tamanio, char palabra[tamanio]
Salida:	La función no retorna valor pero muestra si la palabra es o no palindroma

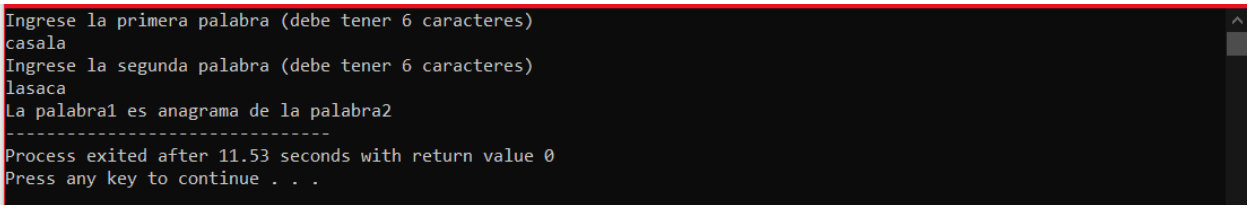
Captura de Pantalla:



7.2 A String is Anagram or Not

Ejercicio:	“A String is Anagram or Not”
Nombre Archivo:	**anagrama.c**
Descripción:	Este programa captura dos strings de tamaño 6 y muestra si la palabra 1 es anagrama de la palabra 2
¿Requiere entrada de datos del usuario?	si
Funciones:	void anagrama(char palabra1[],char palabra2[]) char* contiene(char letra,char palabra[],int tamanio)
Entrada:	char palabra1[6],palabra2[6]
Salida:	Es anagrama o no es anagrama

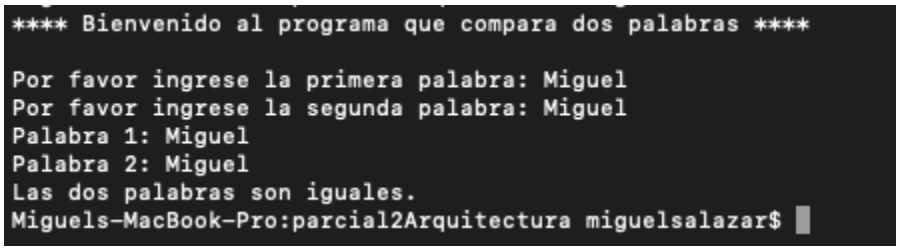
Captura de Pantalla:



7.3 Compare Two Strings

Ejercicio:	“Compare Two Strings”
Nombre Archivo:	**comparaStrings.c**
Descripción:	En este programa, mediante la utilización de una estructura de control for, compara índice a índice los elementos que componen los dos strings introducidos por el usuario, y verifica si son iguales o no.
¿Requiere entrada de datos del usuario?	Sí.
Función Principal:	void comparaStrings(int n, char string1[ ], char string2[ ]);
Entrada:	int n, char string1[ ], char string2[ ]
Salida:	bool iguales

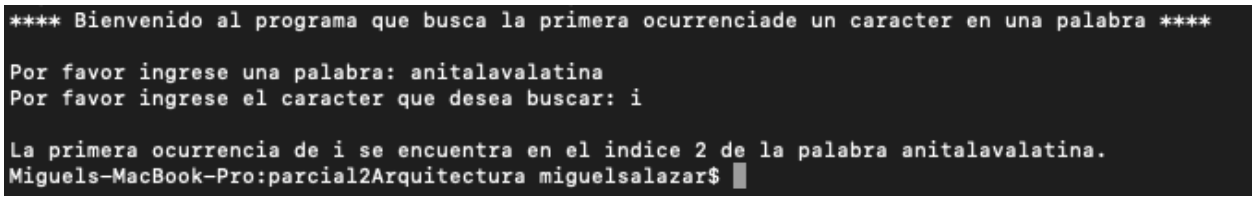
Captura de Pantalla:



Sección 8: “String Questions: Level Up”
8.1 Find The First Occurrence of a Given Character in a Given String

Ejercicio:	“Find The First Occurrence of a Given Character in a Given String”
Nombre Archivo:	**ocurrenciaString.c**
Descripción:	En este programa, mediante la utilización de una estructura de control for, se compara el caracter ingresado por el usuario con todos los elementos (índice a índice) que componen el string
¿Requiere entrada de datos del usuario?	Sí
Función Principal:	void primeraOcurrencia(char c, char palabra[ ])
Entrada:	char c, char palabra[ ]
Salida:	void

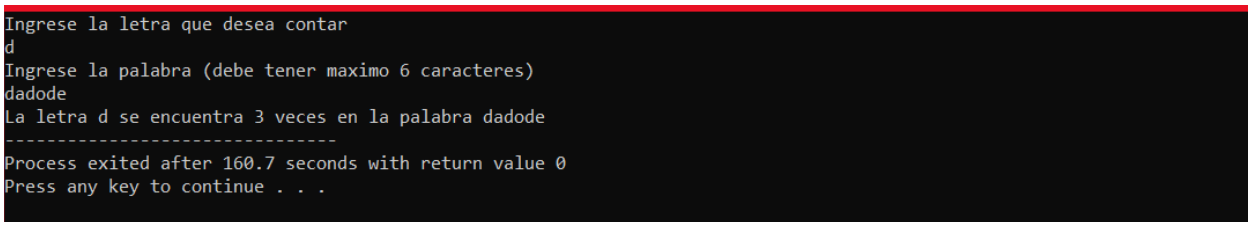
Captura de Pantalla:



8.2 Count Occurrences of a Character in a GIven String

Ejercicio:	“Count Occurrences of a Character in a Given String”
Nombre Archivo:	**conteoCaracter.c**
Descripción:	Este programa cuentan y muestra la cantidad de veces que aparece un caracter ingresado por el usuario en un string de 6 caracteres capturado del mismo modo
¿Requiere entrada de datos del usuario?	si
Función Principal:	int conteo(char palabra[],char letra,int tamanio)
Entrada:	char palabra[6], letra int tamanio
Salida:	int cantidad

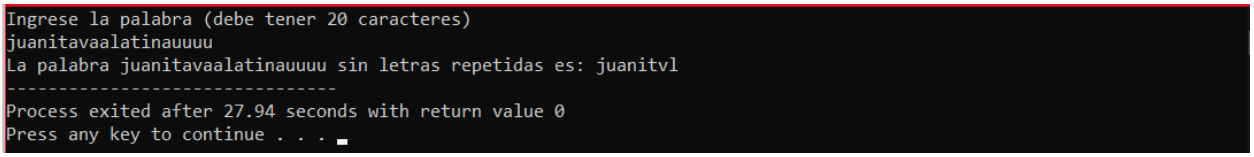
Captura de Pantalla:



8.3 Remove All Repeated Characters From a Given String

Ejercicio:	“Remove All Repeated Characters From a Given String”
Nombre Archivo:	**RemoverRepetidos.c**
Descripción:	Este programa lee un arreglo de 20 caracteres ingresado por el usuario y lo muestra sin ningún carácter repetido
¿Requiere entrada de datos del usuario?	si
Función Principal:	void remover(char palabra[],char p[],int tamanio) int contiene(char letra,char palabra[],int tamanio)
Entrada:	char palabra[20]
Salida:	char palabra[] (sin caracteres repetidos)

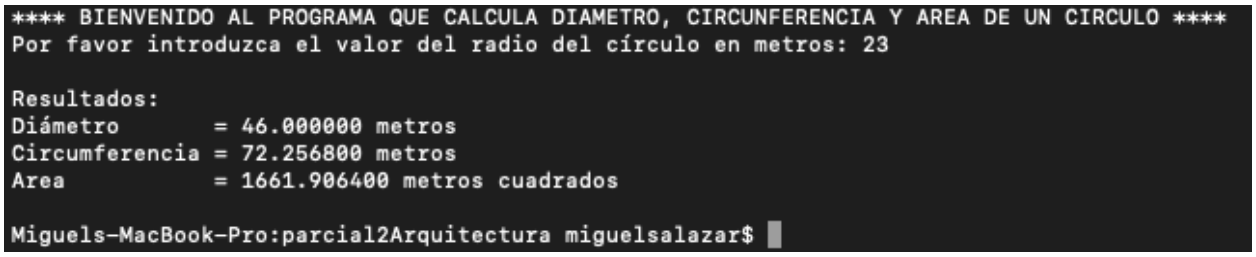
Captura de Pantalla:



Sección 9: “Function Questions”
9.1 Find Diameter, Circumference and Area of a Circle Using Functions

Ejercicio:	“Find Diameter, Circumference and Area of a Circle Using Functions”
Nombre Archivo:	**funcionesCirculo.c**
Descripción:	En este programa, mediante el uso de funciones, se calculan el diámetro, el área y el perímetro de un círculo cuyo radio es ingresado por el usuario. Toma como base las fórmulas básicas de la circunferencia, que son: $d = 2 * r$ $P = 2\pi * r$ $A = \pi * r^2$
¿Requiere entrada de datos del usuario?	Si.
Funciones:	double diametro(double radio) double circunferencia(double radio) double area(double radio)
Entrada:	double radio
Salida:	double diametro, double circunferencia, double area

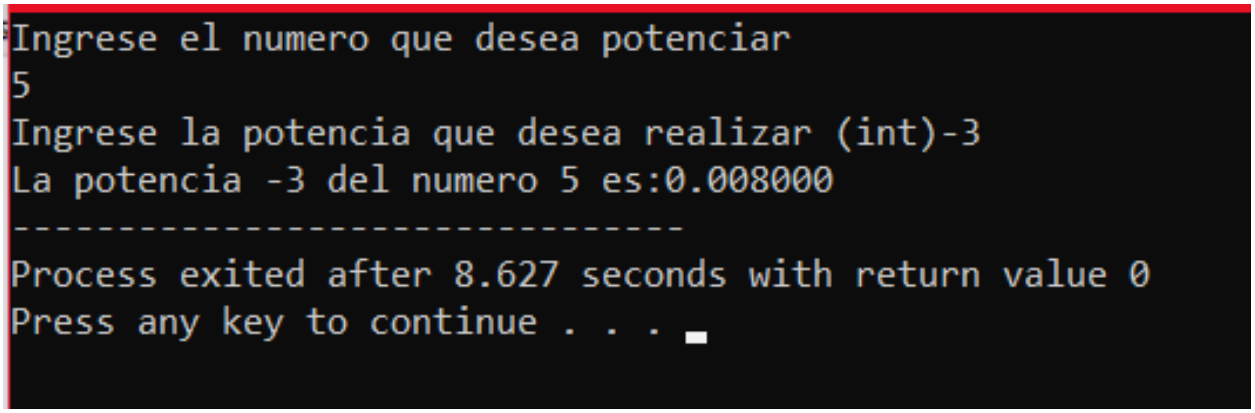
Captura de Pantalla:



9.2 Find The Power of Any Number Using Recursion

Ejercicio:	“Find The Power of Any Number Using Recursion”
Nombre Archivo:	**potenciaRecursiva.c**
Descripción:	Este programa calcula la potencia entera ingresada de un número entero de manera recursiva
¿Requiere entrada de datos del usuario?	si
Función Principal:	int potenciaR(int numero,int potencia);
Entrada:	int numero,potencia
Salida:	int resultado

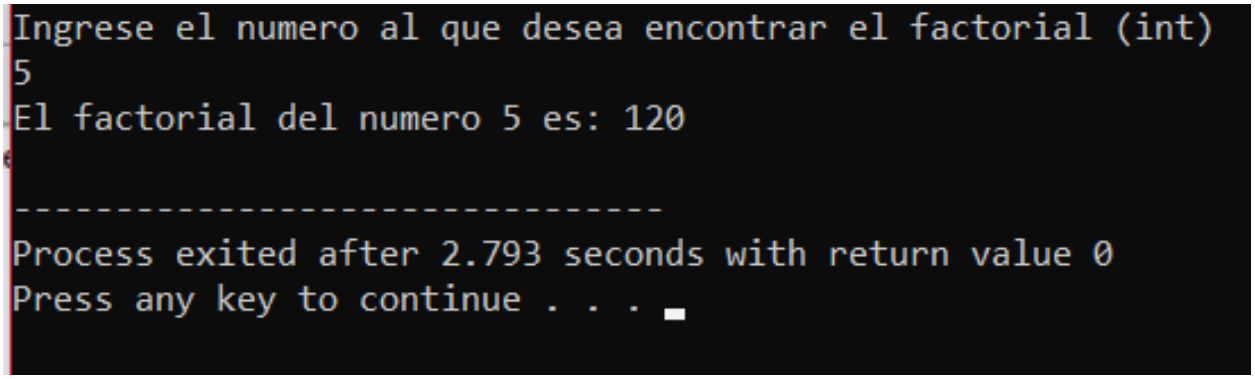
Captura de Pantalla:



9.3 Find The Factorial of Any Number Using Recursion

Ejercicio:	“Find The Factorial of Any Number Using Recursion”
Nombre Archivo:	**factorialRecursivo.c**
Descripción:	Este programa calcula el factorial un número entero positivo de manera recursiva
¿Requiere entrada de datos del usuario?	si
Función Principal:	int factorial(int numero)
Entrada:	int numero
Salida:	int resul

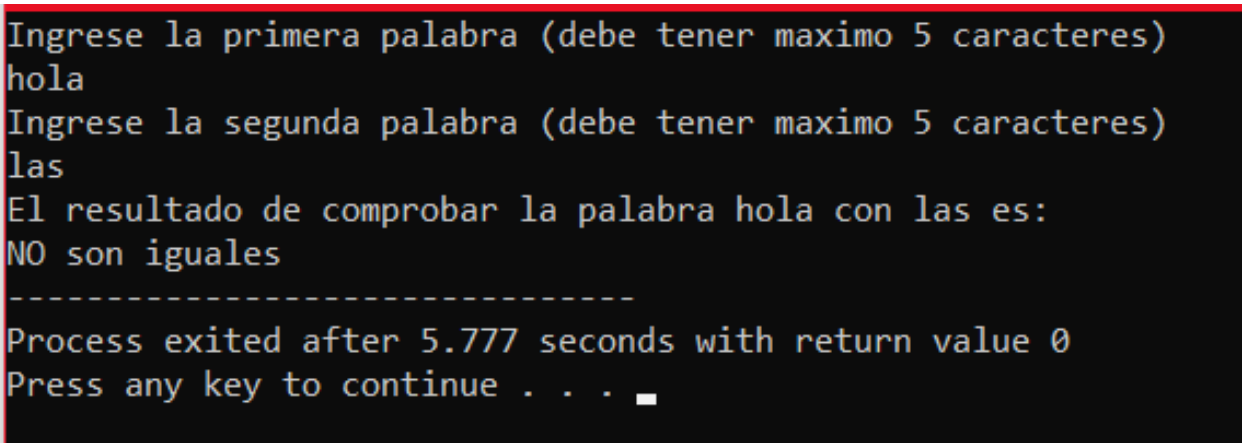
Captura de Pantalla:



Sección 10: “Pointer Questions”
10.1 Compare Two Strings using pointers

Ejercicio:	Compare Two Strings using pointers
Nombre Archivo:	**comprobarStrings.c**
Descripción:	Este programa comprueba que dos strings de máximo 5 caracteres, ingresados por el usuario sean iguales o no, empleando apuntadores.
¿Requiere entrada de datos del usuario?	si
Función Principal:	void verificar(char palabra1[],char palabra2[])
Entrada:	char palabra1[],palabra2[]
Salida:	void “muestra si son o no la misma palabra”

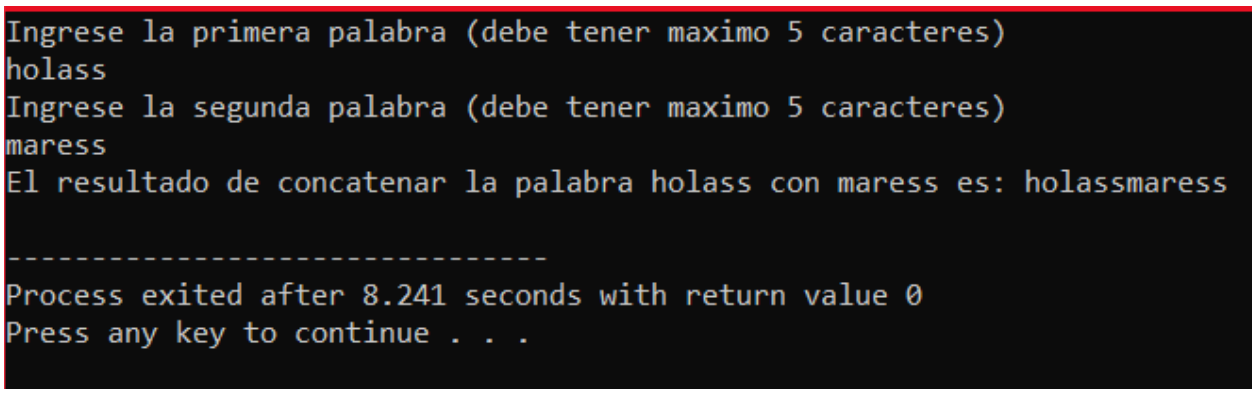
Captura de Pantalla:



10.2 Concatenate Two Strings using pointers

Ejercicio:	Concatenate Two Strings using pointers
Nombre Archivo:	**concatenarApuntadores.c**
Descripción:	Este programa concatena dos strings ingresados por el usuario usando apuntadores y muestra su resultado
¿Requiere entrada de datos del usuario?	si
Función Principal:	void concatenar(char palabra1[],char palabra2[])
Entrada:	char palabra1[],palabra2[]
Salida:	void “muestra el resultado de la concatenación”

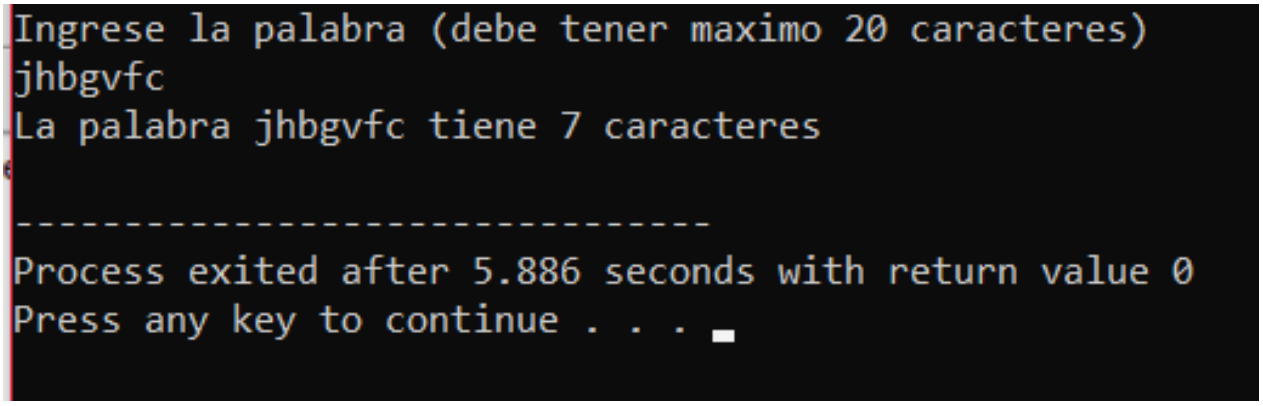
Captura de Pantalla:



10.3 Find the Length of the String using pointers

Ejercicio:	Find the length of the string using pointers
Nombre Archivo:	**tamanioArregloApuntadores.c**
Descripción:	Este programa cuenta la cantidad de caracteres ingresados por el usuario en un string, usando apuntadores
¿Requiere entrada de datos del usuario?	si
Función Principal:	int conteo(char palabra1[])
Entrada:	char palabra1[]
Salida:	int i “cantidad de caracteres”

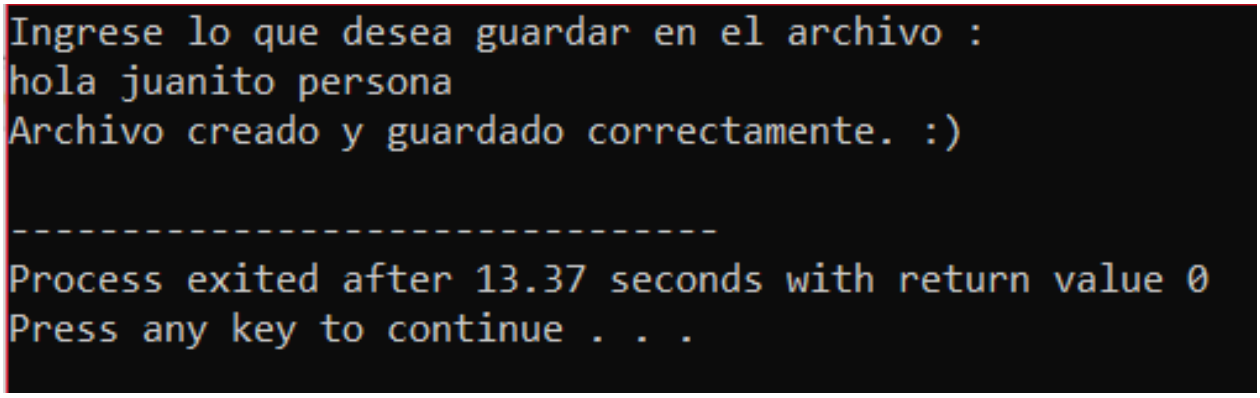
Captura de Pantalla:

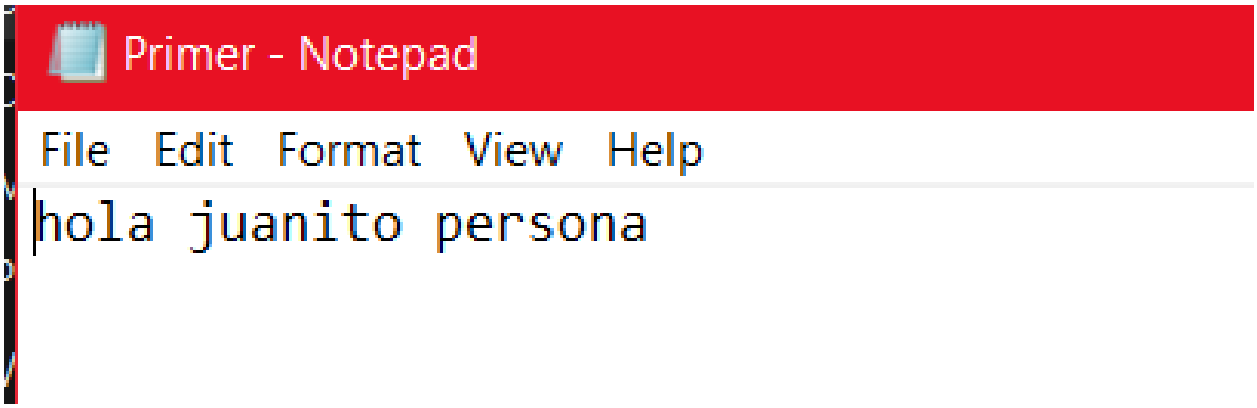


Sección 11: File Handling
11.1 Create a File and Write contents, Save and close the file

Ejercicio:	Create a file and write contents, save and close the file
Nombre Archivo:	**manejoArchivos.c**
Descripción:	Este programa crea un archivo txt con nombre: Primer escribe en él lo cierra y guarda
¿Requiere entrada de datos del usuario?	si
Función Principal:	main()
Entrada:	char data[] “información a escribir”
Salida:	“Fue guardado correctamente” -- “no se creó el doc”

Captura de Pantalla:





11.2 Read File Contents and Display Them On The Console

Ejercicio:	“Read File Contents and Display Them On The Console”
Nombre Archivo:	**leerTexto.c**
Descripción:	Este programa, mediante la utilización de un puntero (FILE *), se lee un archivo de texto línea por línea. Cada línea se carga a un arreglo de caracteres que se imprime mediante la utilización de una estructura de control tipo While.
¿Requiere entrada de datos del usuario?	No.
Entrada:	archivo.txt
Salida:	char[ ]

Capturas de Pantalla:

1. Archivo de texto .t

archivoDeTexto.txt

```
1   Inicio.
2   Este es un archivo de texto.
3   Me van a utilizar para la sección 11 del parcial.
4   Mis autores son Miguel Salazar y Diego Bermúdez.
5   Fin.
```

2. Ejecución del programa:

```
**** Bienvenido al programa que lee archivos de texto. ****

EL texto escrito en el archivo archivoDeTexto.txt es:
Inicio.

Este es un archivo de texto.

Me van a utilizar para la sección 11 del parcial.

Mis autores son Miguel Salazar y Diego Bermúdez.

Fin.
Miguels-MacBook-Pro:parcial2Arquitectura miguelsalazar$
```

11.3 How to Copy Contents From one FIle to Another

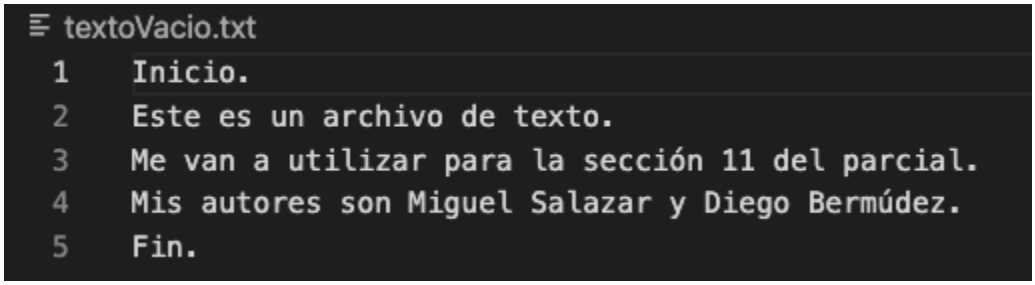
Ejercicio:	“How To Copy Contents From One File to Another”
Nombre Archivo:	**copiarContenido.c**
Descripción:	En este programa se utilizan dos punteros: uno para la lectura del archivo de texto, y otro para la escritura del mismo texto en otro archivo. Se carga el texto del archivo 1 a un arreglo de caracteres y ese mismo arreglo se utiliza para copiarlo en el archivo 2.
¿Requiere entrada de datos del usuario?	No.
Entrada:	Archivo .txt a char[ ]
Salida:	char[ ] a Archivo .txt

Captura de Pantalla:

1. Ejecución del programa:

```
**** Bienvenido al programa que lee un archivo de texto y lo escribe en otro archivo vacío ****

EL texto escrito en el archivo archivoDeTexto.txt es:
Inicio.

Este es un archivo de texto.

Me van a utilizar para la sección 11 del parcial.

Mis autores son Miguel Salazar y Diego Bermúdez.

Fin.
Revisa ahora el archivo con nombre textoVacio.txt que se encuentra en este fichero.
Miguels-MacBook-Pro:parcial2Arquitectura miguelsalazar$
```

2. Resultado en el archivo textoVacio.txt:



Sección 12: Sorting
12.1 Bubble Sort in C

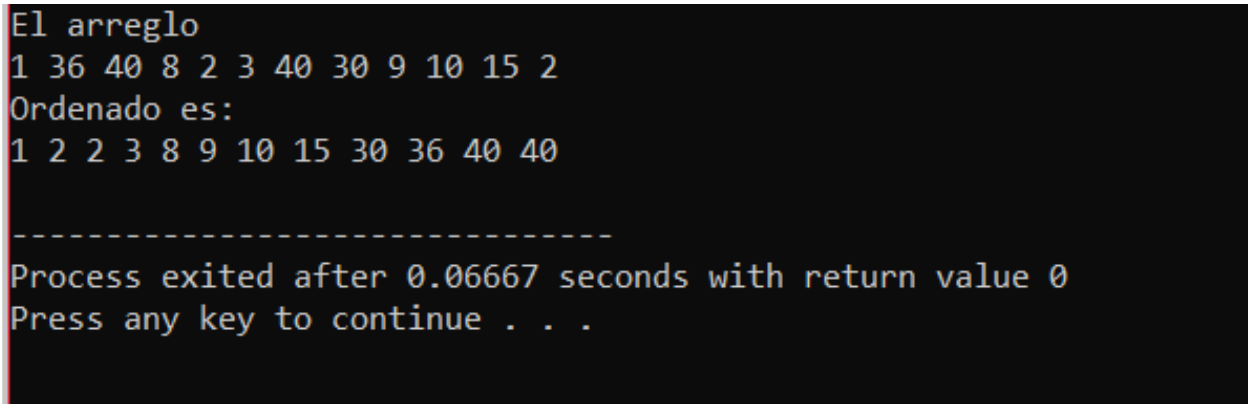
Ejercicio:	“Bubble Sort in C”
Nombre Archivo:	**bubbleSort.c**
Descripción:	En este programa se implementa el algoritmo clásico denominado “Bubble Sort”, con dos de sus mejoras implementadas. Su funcionamiento básico consta en revisar cada elemento de la lista con su vecino derecho e intercambiarlos en el caso de que aquel de la izquierda sea mayor que el de la derecha.
¿Requiere entrada de datos del usuario?	No.
Función Principal:	void bubbleSort(int arreglo[ ], int tamanio)
Entrada:	int arreglo[ ], int tamanio
Salida:	void

Captura de Pantalla:

12.2 Insertion Sort In C

Ejercicio:	“Insertion Sort in C”
Nombre Archivo:	**insertionSort.c**
Descripción:	Este programa ordena un arreglo de enteros predeterminado mediante el método de inserción
¿Requiere entrada de datos del usuario?	no
Función Principal:	void sortInsertion(int numeros[],int tamanio)
Entrada:	int numeros[]
Salida:	void “muestra el arreglo ordenado”

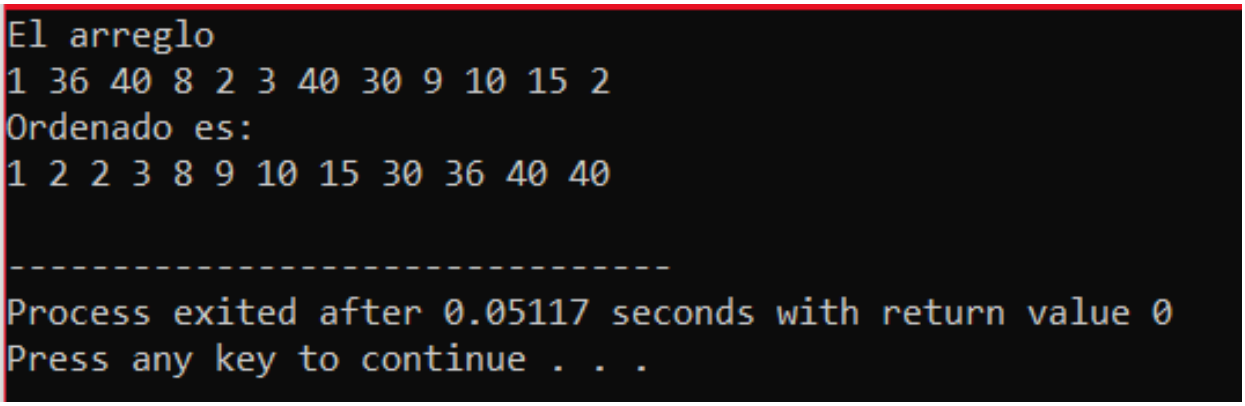
Captura de Pantalla:



12.3 Selection Sort in C

Ejercicio:	“Selection Sort in C”
Nombre Archivo:	**selectionSort.c**
Descripción:	Este programa ordena un arreglo de enteros predeterminado mediante el método de seleccion
¿Requiere entrada de datos del usuario?	no
Función Principal:	void sortSeleccion(int numeros[],int tamanio)
Entrada:	int numeros
Salida:	void “muestra el arreglo ordenado”

Captura de Pantalla:

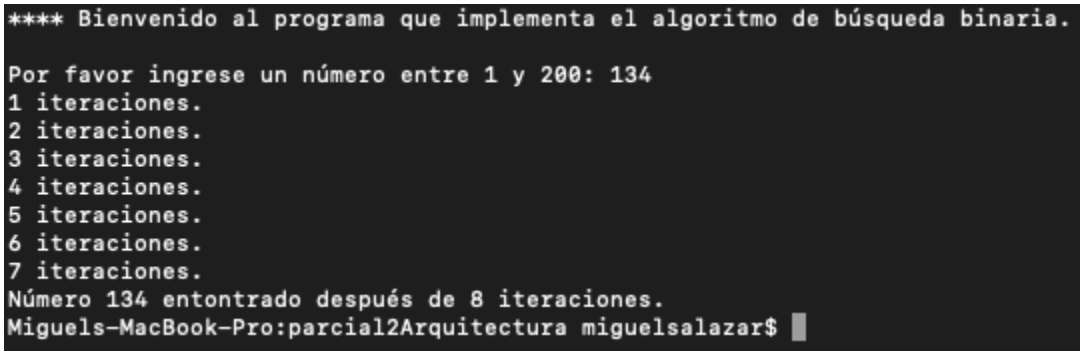


Sección 13: Searching
13.1 Binary Search

Ejercicio:	Binary Search
Nombre Archivo:	**binarySearch.c**
Descripción:	En este programa se implementa el algoritmo clásico de búsqueda binaria, que implementa una estrategia conocida como “divide y vencerás”, mediante la cual se divide la estructura principal en la mitad, y subsecuentemente en la mitad de la mitad y así sucesivamente. Solamente se puede utilizar en estructuras que ya se encuentran ordenadas.
¿Requiere entrada de datos del usuario?	Sí.
Función Principal:	void busquedaBinaria(int array[ ], int indexInicio, int indexFinal, int num)
Entrada:	int array[ ], int indexInicio, int indexFinal, int num

Salida:	int

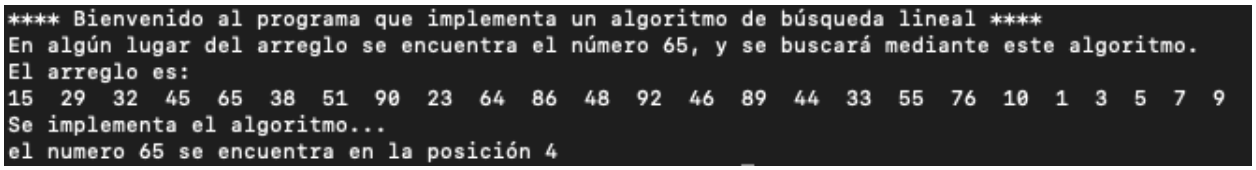
Captura de Pantalla:



13.2 Linear Search

Ejercicio:	Linear Search
Nombre Archivo:	**linearSearch.c**
Descripción:	En este programa, por medio de la utilización de una estructura de control for, se efectúa una búsqueda lineal del número 65, que se encuentra ubicado en algún índice del arreglo.
¿Requiere entrada de datos del usuario?	No.
Función Principal:	void busquedaLineal(int n, int tamanio, int arreglo[n]);
Entrada:	int n, int tamanio, int arreglo[ ]
Salida:	void

Captura de Pantalla:



13.3 Recursive Binary Search in C

Ejercicio:	Recursive Binary Search in C
Nombre Archivo:	**busquedaBinariaRecursiva.c**
Descripción:	Este programa determina la existencia o ausencia de un numero entero ingresado por el usuario en un arreglo de enteros predeterminado, mostrando la posición en la cual se encuentra el valor
¿Requiere entrada de datos del usuario?	Si.
Función Principal:	void busqueda(int numeros[],int indI,int indF,int x)
Entrada:	int n, int tamanio, int arreglo[ ]
Salida:	int posicion

Captura de Pantalla: