PRÁCTICA DIRIGIDA 4

1. Escriba un programa usando punteros que genere las letras del alfabeto español.

2. Escriba un programa usando punteros que sume los elementos de un arreglo.

3. Escriba un programa que **use apuntadores** para generar un arr[n] con un número pequeño de datos enteros entre 1 y 12, ordenados ascendentemente y que los reporte.

Sugerencia de diseño: genere los números al azar y los ordena.
Ejemplo de salida: 2 5 6 9 10 12

4. Ya resolvimos el problema 3, los datos están ordenados; pero resulta que ahora los necesitan descendentes (o sea en reversa); como hemos perdido tiempo, debemos reordenarlos en el modo más rápido posible. **Use apuntadores**.

Ejemplo de salida:

2 5 6 9 10 12
12 10 9 6 5 2

Sugerencia de diseño: Utilice un solo for() y aplique la enseñanza de Cristo: “Los últimosserán los primeros” y viceversa.

5. Escriba y pruebe una función que reciba una referencia a un arreglo de enteros formado por valores diferentes de cero, salvo el último término y retorne el mayor de los elementos de dichoarreglo. Para ello debe completar en los puntos suspensivos:

```
#include <iostream >
using namespace std;

int maximo(int * p);

int main(){
    int edades[] = {22,19,17,23,18,0};
    printf(" edad máxima: %d\n",
    maximo(edades));
}

int maximo(int * p){/*...*/;}
```

Para el ejemplo, La salida debe ser:

edad máxima: 23

6. Escriba una función tal que al pasarle una referencia a un arreglo de enteros formado enteros diferentes de cero, salvo el último término, muestre todos los elementos distintos de cero de dicho arreglo pero en sentido contrario. También escriba otra función tal que al pasarle una referencia a un arreglo de enteros formado enteros diferentes de cero, salvo el último término,retorne la cantidad de elementos distintos de cero de dicho arreglo. Para ello debe completar en los puntos suspensivos:

```
#include <stdio.h>
void contrario(int * p);
```

```
int longitud(int * p);
int main() {
    int numeros[4] = {-1,-3,-2,0};
    printf(" cantidad de numeros:%3d\n", longitud(numeros));
    contrario(numeros);
}

int longitud(int * p){/*...*/;}
void contrario(int * p){/*...*/;}
```

Para el ejemplo, La salida debe ser:

cantidad de números: 3

-2 -3 -1

7. Escriba una función que reciba una referencia a un arreglo de enteros con al menos un elemento igual a cero, muestre todos los elementos de dicho arreglo hasta antes del primerelemento igual a cero. También escriba otra función tal que al pasarle un par de referenciasa arreglos de enteros con al menos un elemento igual a cero, copie todos los elementos hasta antes del primer elemento igual a cero del arreglo de la segunda referencia en los elementos del arreglo de la primera referencia. Para ello debe completar en los puntos suspensivos:

```
#include <iostream>
using namespace std;

void copiar(int * des, int * ori);
void mostrar(int * p);
int main(){
    int destino[5] = {4,5,6,7,0};
    int origen[4] = {-1,-3,-2,0};
    mostrar(destino);
    copiar(destino, origen);
    mostrar(destino);
}
void mostrar(int * p){/*...*/;}
void copiar(int * des, int * ori){/*...*/;}
```

Para el ejemplo, La salida debe ser:

4 5 6 7

-1 -3 -2 7

8. Escriba una función tal que al pasarle una referencia a un arreglo de enteros formado enteros diferentes de cero, salvo el último término, muestre todos los elementos distintosde cero de dicho arreglo. También escriba otra función tal que al pasarle una referencia a un arreglo de enteros formado enteros diferentes de cero, salvo el último término, retornela cantidad de elementos distintos de cero de dicho arreglo. Luego, implemente una función tal que al pasarle un par de referencias a arreglos de enteros diferentes de cero, salvo el último término, anexe, desde la posición del elemento igual a cero del arreglo referenciado por la primera referencia, todos los elementos del arreglo referenciado por la segunda referencia. Para ello debe completar en los puntos suspensivos:

```
#include <iostream>
```

```
Using namespace std;
void anexar(int * des, int * ori);
void mostrar(int * p);
int longitud(int * p);
int main(){
    int destino[6] =  {4,5,0};
    int origen[4] = {-1,-3,-2,0};
    mostrar(destino);
    anexar(destino, origen);
    mostrar(destino);
}
int longitud(int * p){/*...*/;}
void mostrar(int * p){/*...*/;}
void anexar(int * des, int * ori){/*...*/;}
```

Para el ejemplo, La salida debe ser:

4 5
4 5 -1 -3 -2

9. Escriba la función sumaArreglos() que tiene como entrada dos arreglos de tamaños **m** y **n** y retorna la suma de los elementos de ambos arreglos. Desde la main() defina, asigne valores, ordene la suma, e imprima todo lo importante.

10. Escriba la función sumaArreglo() que tiene como entrada tres arreglos de tamaño **n**, sumelos valores de los elementos de los dos primeros y colóquelos en el tercero. Desde la main() defina, asigne valores, ordene la suma, e imprima todo lo importante.

11. Escriba la función productoArreglo() que tiene como entrada tres arreglos de tamaño **n**x **n** de entrada, multiplique los dos primeros y colóquelo en el tercero. Desde la main() defina, asigne valores, ordene la multiplicación, e imprima todo lo importante.

12. Escriba la función:

```
void asignar(int n, int *arr1, int *arr2)
```

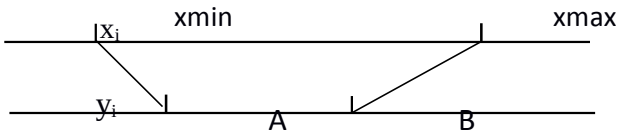
Donde **n** es el número de elementos de arr1 y arr2; asignar() carga los datos apuntados por arr1 en los apuntados por arr2 en el siguiente orden:

0 → 0
n-1 → 1
1 → 2
n-2 → 3
2 → 4
…

Ejemplo: arr1[5] = {10, 20, 30, 40, 50}
arr2[5] = {10, 50, 20, 40, 30}

Desde la main() defina, asigne valores, ordene la carga e imprima todo lo importante.

13. Normalización de datos a un intervalo [A, B]: Dado un montón de puntos xi que se distribuyen en un intervalo [xmin, xmax] ≡ [mínimo de x, máximo de x] que se guardan en un arreglo x, se requiere reubicarlos, como yi en un intervalo [A, B] y guardarlos en un arreglo y, los yi guardan la proporción de la separación de los xi



Los y_i se calculan así:

$$y_i = ax_i + b$$

Donde:

a = (B-A) / (xmax – xmin) b = A - a*xmin

Utilice apuntadores px, py que apuntan a x respectivamente, defina **x[n]**, n=10 y asígnele elementos aleatorios x_i entre 2 y 20. Calcule **a** y **b** para un intervalo [A, B] = [-1, 1]; defina **y[n]** y asígnele valores y_i :

$$y_i = \mathbf{a}x_i + \mathbf{b}$$

Reporte x, y. La salida puede ser:
Normalización de datos:
Inicio: 13.00 16.00 6.00 14.00 **20.00** 15.00 15.00 **2.00** 14.00 10.00
Fin : 0.22 0.56 -0.56 0.33 **1.00** 0.44 0.44 **-1.00** 0.33 -0.11

14. Programe la función

void carga(int *arr, int n)

La cual cargará números aleatorios entre 10 y 60; pero a medida que ingresa los ordena ascendentemente, ejemplo:

Aleatorio	arr
46	**46**
67	46, **67**
25	**25**, 46, 67
32	25, **32**, 46, 67

15. Regresión lineal: Dado una gran población de pares de puntos (xi, yi), se pueden graficar en R2:

Se puede ejecutar un proceso de "regresión lineal" para obtener la recta: y = ax+b "de mejor ajuste" a la nube de puntos, los valores de a y b se calculan así:

mx = media de los x_i; my = media de los yi

$a = \sum(xi-mx)(yi-my) / \sum(xi-mx)2$ b = my - a*mx

Escriba un programa que guarde los (x_i, y_i) en dos arreglos x, y de tamaño **n=40**. **Utilice apuntadores** px, py, que apuntan a x, y respectivamente:

Sugerencia de desarrollo:

```
...
for(i=0, px=x, py=y, mx=0, my=0; i<n; i++, px++, py++){
*px = rand()%50+1;                                   // genera xi
*py = 3 * *px + 2 + (rand()%1000+1)/2000-0.025; // genera yi: a y b serán ≈ 3 y 2 mx
= … // suma de los xi
my = …                                   // suma de los yi
}
...
```

Calcule a y b y repórtelos. Un ejemplo de salida es: Regresión lineal:

y = ax + b
y = **3.00**x + **1.98** // a ≈ 3, b ≈ 2

16. Defina una función con prototipo: void invertir(int* p) tal que al recibir el nombre de un arreglo de enteros de elementos distintos de cero, salvo el ultimo, imprima la cantidad de elementos diferentes de cero y todos los elementos distintos de cero del arreglo desde aquel de mayor ´índice hasta el de índice cero (¡en ese orden!). Finalmente, pruebe su desarrollo definiendo la función principal como:

```
int main() {
   int arr1[5] = {1,2,3,4,0};
   int arr2[10] = {1,2,3,4,5,6,7,8,9,0};
   invertir(arr1);
   invertir(arr2);
}
```

17. Usando funciones, punteros y vectores desarrollar un programa donde se tiene el siguiente vector tipo float {16.6, 35.9, 5.7, 25.9, 5.1, 9.0} encontrar el promedio y luego contar cuantos números son mayores al promedio y cuantos son menores del promedio.

Ejemplo

Imprimir array:
16.6 135.9 5.7 25.9 5.1 9
el promedio es: 33.033
mayores: 1 menores: 5

18. Defina una función con prototipo: void concatenar(int* p1, int* p2) tal que al recibir el nombre de dos arreglos de enteros de elementos distintos de cero, salvo el último, imprima loselementos distintos de cero del primer arreglo seguido de los elementos distintos de cero del segundo. Finalmente, pruebe su desarrollo definiendo la función principal como:

```
int main() {
   int arr1[5] = {1,2,3,4,0};
   int arr2[10] = {1,2,3,4,5,6,7,8,9,0};
   concatenar(arr1,arr2);
   concatenar(arr2,arr1);
}
```

19. Escriba la función:

*void cargar(int n, int *array1, int *array2)*

Donde **n** es el número de elementos de array1 y array2; y ***cargar()*** asigna los datos apuntados por array1 en los apuntados por array2 en el siguiente orden:

0 → 0
n-1 → 1
1 → 2
n-2 → 3
2 → 4

...

Ejemplo:

Si la entrada es:

n: 8
array1: 15 25 35 45 55 65 75 85

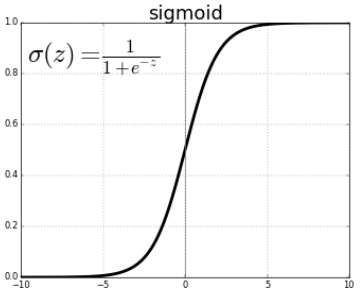
la salida será:

array2: 15 85 25 75 35 65 45 55

Desde la función main() defina, asigne valores, ordene la carga e imprima.

20. La regresión logística es una técnica de clasificación binaria de datos, el modelo se entrenó con los datos mostrados en la tabla obteniendo los siguientes parámetros del modelo, **w = 1.2020** y **b = -3.4443**. La tabla de datos es:

Horas de Estudio (x)	Aprueba? (y)	Predicción (pred)
0.50	0	0
0.75	0	0
1.00	0	0
1.25	0	0
1.50	0	0
1.75	0	0
1.75	1	0
2.00	0	0
2.25	1	0
2.50	0	0
2.75	1	0
3.00	0	0
3..25	1	0
3.50	0	0
4.00	1	0



Se desea determinar la precisión de estos parámetros, para lo cual se deben implementar las siguientes funciones:

void sigmoidea(double *p);

Esta función recibe un vector y lo modifica sacando la función sigmoidea de cada elemento.
Ejm: entrada [0,1,0,1] => salida [1,0.73,1.0.73]]
para el valor 0 => f(0) = 1/(1+ e^0) = 1. Usar e = 2.72.

void producto(double w, double *p);
Recibe un vector al cual se le multiplica por un decimal a cada uno de sus elementos
Ejm: entrada (2 y [2,3,4,5]) => salida [4,6,8,10], tenga en cuenta que el vector de entrada es el que se modifica.

void suma(double b, double *p);

Recibe un vector al cual se suma un valor a cada uno de sus elementos.
Ejm: entrada (2 y [2,3,4,5]) => salida [4,5,6,7], tenga en cuenta que el vector de entrada es el que se modifica.

void fija(double *p, double *q);
Recibe dos vectores, si *p > 0.5 -> *q = 1 en caso contrario *q = 0.
Ejm: entrada ([0.2,0.6,0.7], [0,0,0]) => valores modificados de *q => [0,1,1]

int compara(double *p, double *q);
Compara los dos vectores y retorna la cantidad de coincidencias
Ejm: entrada ([1,0,1,0],[1,1,1,1]) => salida 2

Se sugiere el siguiente código:

```
int main(){

// definir : vector x,y       con los datos de la tabla mostrada.
// definir : w, b.            datos dados en el enunciado.
// definir : vector pred      inicializar a cero.
producto(w,x);
suma(b,x);
sigmoidea(x);
fija(x,pred);
cout<<compara(y,pred)<<endl;
 }
```