	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Profesor:	Tista Garcia Edgar
Asignatura:	EDA II
Grupo:	5
No de Práctica(s):	3
Integrante(s):	Calzada Martinez Jonathan Omar
No. de Equipo de cómputo empleado	35
Semestre:	2019-2
Fecha de entrega:	25/02/2019
Obervaciones:	

CALIFICACIÓN: _____

Objetivo:

El estudiante identificará la etructura de los algoritmos de ordenamiento merge sort, counting sort y radix sort

Introducciòn:

Los algoritmos de ordenamiento nos permite, como su nombre lo dice, ordenar información de una manera especial basándonos en un criterio de ordenamiento.

En la computación el ordenamiento de datos cumple un rol muy importante, ya sea como un fin en sí o como parte de otros procedimientos más complejos. Se han desarrollado muchas técnicas en este ámbito, cada una con características específicas, y con ventajas y desventajas sobre las demás.

1)Counting sort

Ejercicio 1 counting sort

Se realizó el arreglo de 25 enteros como se pidio en la practica, piendo los datos al usuario esperando que metio los datos correctos (aun que se le puso una codicional para verificar si es correcto o no)

```
for(i =1;i<=k;i++){
    arr2[i]=arr2[i]+arr2[i-1];
//printf("%d",arr2[i]);
}


for(j=n;j>=1;j--)
{
    arr1[arr2[arr3[j]]] = arr3[j];
    arr2[arr3[j]] = arr2[arr3[j]] - 1;
}
printf("\nEl arreglo ordenado es : ");
for(i=1;i<=n;i++)
{
    printf("\t");
    printf("%d",arr1[i]);
}
```

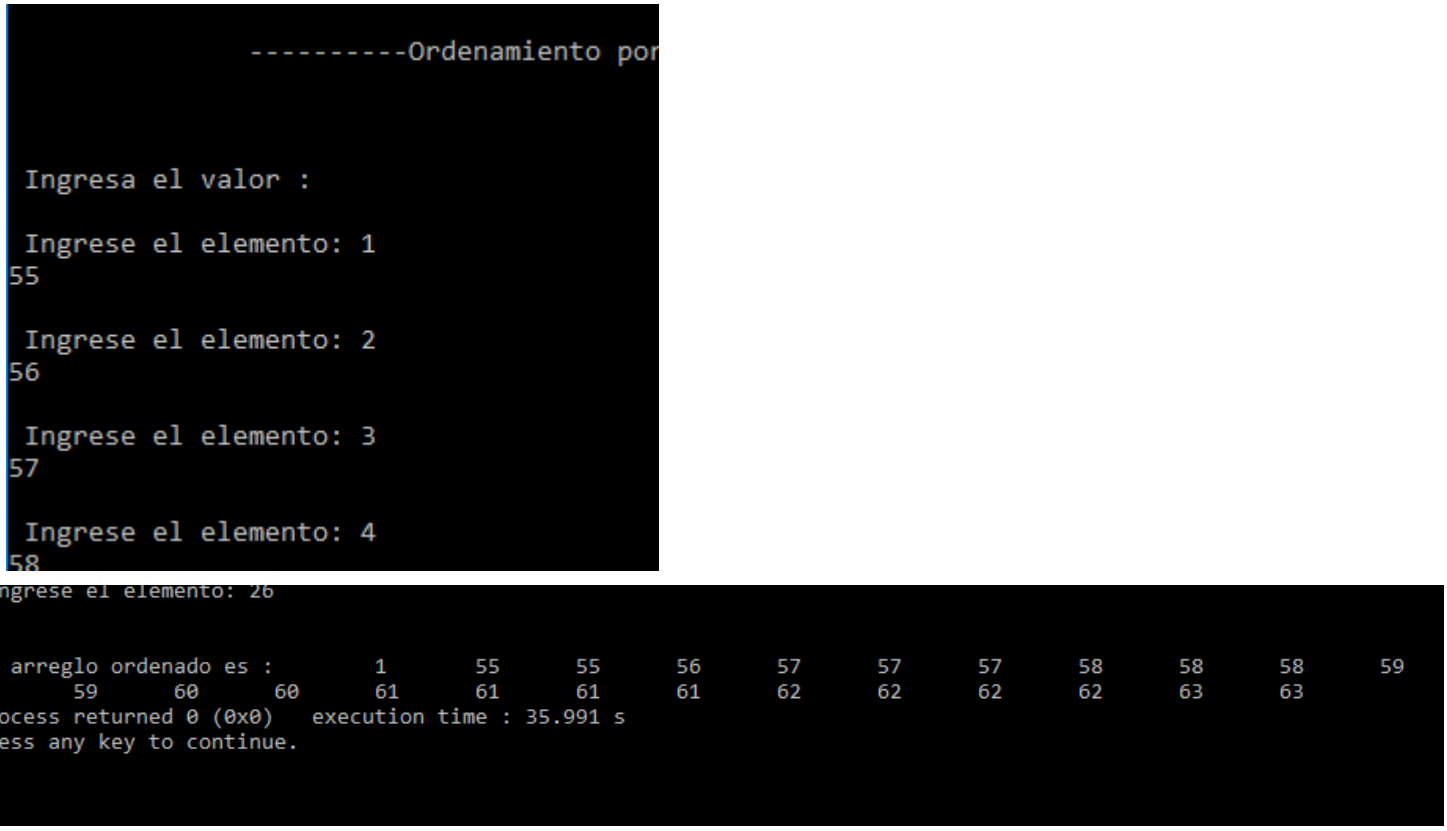
En este apartado del codigo se realizaron las tres for para realizar el conteo de los numeros repetidos, en la lista de numeros.

```
    printf("\n Ingrese el elemento: %d \n",i+1);
    scanf("%d",&num);
    // num = rand () % 11 + 20;;
    if((num>=55)&&(num<=63)){
    arr3[i]=num;
    if(arr3[i] > k)
    {
        k = arr3[i];
    }

lse{
        printf("\n Valor invalido \n");
        i=i-1;
```

ordenamiento en el arreglo de salida.

Al ingresar los elementos, La pantalla de salida se muestra en la parte inferior.



Analisis:

El algoritmo de couting sort se veía más sencillo de lo que me costo al hacer el codigo, el problema fue a la hora de hacer el ultimo arreglo ordenado ahí me atore bastante no sabía como realizar ese paso. Así que me vi en la necesidad de orientarme en internet y encontre un pseudocodigo que me ayudo bastante al acomodar el orden el algoritmo (la imagen se encuentra abajo en las conclusiones)

2) Radix Sort

Para realizar el algoritmo de ordenamiento Radix se le realizaron colas para poder hacer el ordenamiento, así que se creo una estructura llamada nodo.

Se pide al usuario que ingrese los datos así como el numero de datos que se va ingresar.

```
printf(" Ingresa los elementos %d : ",conta+1);
scanf("%d",&num_item);

temp= malloc(sizeof(struct nodo));
temp->dato=num_item;
temp->sig=NULL;

if(inicio==NULL)
    inicio=temp;
else
{
    q=inicio;
    while(q->sig!
        q=q->sig;
    q->sig=temp;
}
```

```
int masSignificativo()
{
    struct nodo*p=inicio;
    int large=0,ndigit=0;

    while(p != NULL)
    {
        if(p ->dato > large)
            large = p->dato;
        p = p->sig ;
    //printf("%d",large[p->next]);
    }

    while(large != 0)
    {
        ndigit++;
        large = large/10 ;
```

Se realizo una funcion para ver cual es el elemento mas significativo de los valores ingresados de esta manera va abanzando 1 elemento en cada cada iteración que se hace

Anlisis RADIX SORT:

El codigo que realice no funciona ya que tuve problemas al realizar la funcion para hacer el conteo de datos en las colas y no pude implementarla, según yo teniendo esa funcion tendría que funcionar bien mi codigo pero lamentablemente no fue así.
Comparando este con los otros algoritmos este se me hizo más complicado al hacerlo en codigo.

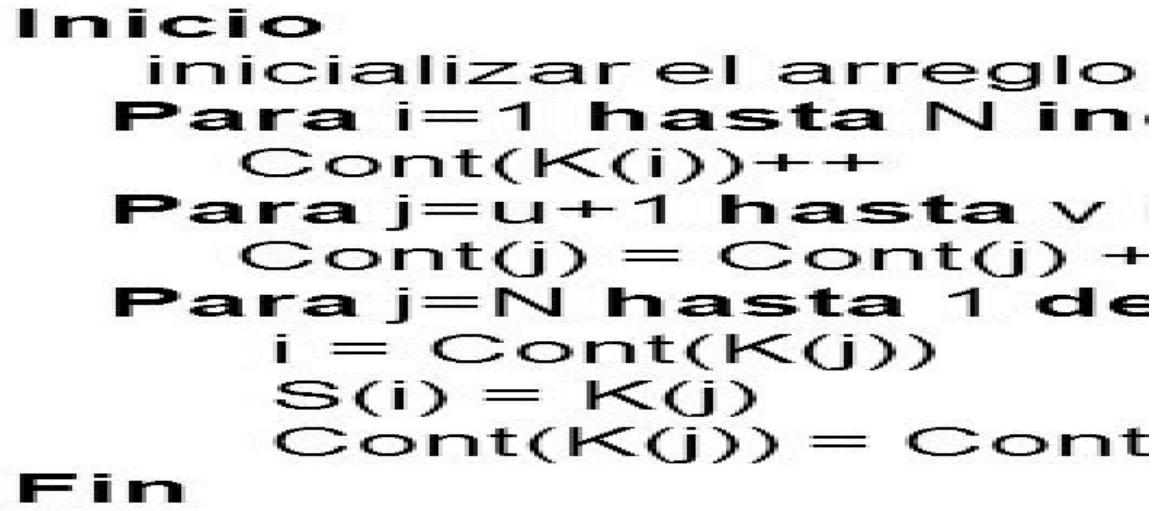
Ejercicio 3:
Analisis:
El ejercicio 3 por falta de tiempo y de experiencia en java no pude realizar aun que sea un poco el ejercicio 3 de mergesort.
Merge sort con lo visto en clase se me hizo algo sencillo almenos en la teoria, ya que se trata de partir los arreglos en dos y esos dos partirlos en dos, luego al juntarlos van a estar ordenados. me hizo falta tiempo y estudiar java para poder realizar el ejercicio 3.

Conclusiones.

En esta practica tuve dificultades al realizarla por el motivo de que no recordaba bien el funcionamiento de el algoritmo de ordenamiento por conteo entonces estaba algo confundido en como debìan ir las cosas. Asì mismo tambièn me encontre con dificultades con rcordar algunas sintaxis de el lenguaje C ya que habìa dejado de practicarlo por lo que me atoro unos minutos.

Lo que alcance a terminar fue los dos arreglos y el conteo de los numeros por separado, solo me entretube tratando de que me aparezca el numero y la cantidad de veces que se repite a la hora de imprimir en pantalla.

Para el primer ejercicio que es counting Sort se logro realizar la practica aun que tuve que borrar el codigo que había realizado en el laboratorio debedo a que no hallaba la forma de realizar el ordenamiento en el arreglo final así que me vi la necesidad de orientarme de un pseudocodigo que encontre, lo dejo en seguida



https://ccc.inaoep.mx/~jagonzalez/ADA/Ordenamiento-b.pdf

Este pseudo codigo que encontre me fue de mucha ayuda.

Para el ejercicio 2 tuve muchas dificultades y más a la hora de implementar la funcion de conteo de datos que hay en las colas.

Para el tercer ejercicio no me dio tiempo de terminarlo ya que se me juntaron las cosas. A parte de que tenia ciertas dudas en la implementación.

La parte teorica sí me quedo clara pero el problema fue al realizar en el codigo, con esto me doy cuenta que tengo mucho que estudiar y practicar C y java.