UNIVERSIDAD AUTONOMA GABRIEL RENE MORENO

FACULTAD DE INGENIERÍA Y CIENCIAS DE LA COMPUTACIÓN Y TELECOMUNICACIONES

ESTRUCTURA DE DATOS II

CONTENIDO: REPRESENTACIÓN DE LISTAS EN ARREGLO.

Implementar en Lenguaje JAVA.

PORCENTAJE TERMINADO: 100%

GRUPO: 14

INTEGRANTES	DT	HG	HI	EVAL
Flores Veizaga Eudenia Gandira	2	1	1	100
Garcia Taboada Brayan Albaro	2	1	1	100
Gedalge Cayhuara Cristian Gabriel	2	1	1	80
Haquin Serrano Rodrigo	2	1	1	90
Hernandez Lijeron Roly	2	1	1	100

//DT, días trabajados

//HG, horas grupo

//HI, horas individual

Fecha de presentación : Lunes, 18 de Marzo de 2024

Fecha Presentada : Lunes, 18 de Marzo de 2024

Días de Atraso : 0

REPRESENTACIÓN DE LISTAS EN ARREGLO.

Implementar en Lenguaje JAVA.

TRABAJO EN GRUPOS.

1. L1.insertarIesimo(x, i) : Método que inserta el elemento x, en la posición i, de la lista L1.

```
public void insertarIesimo(int x, int i){
    int k = this.cant-1;
    while (k>=i){
        this.elem[k+1]=this.elem[k];
        k=k-1;
    }
    this.elem[i]=x;
    this.cant++;
}
```

2. L1.insertarPrim(x) : Método que insertar el elemento x, al inicio de la lista L1.

```
public void insertarPrim(int x){
    insertarIesimo(x,0);
}
```

3. L1.insertarUlt(x) : Método que inserta el elemento x, al final de la lista L1.

```
public void insertarUlt(int x){
    insertarIesimo(x, this.cant);
}
```

4. L1.eliminarIesimo(i) : Método que elimina el elemento de la posición i.

```
public void eliminarIesimo(int i){
    int k=i+1;
    while (k<this.cant){
        this.elem[k-1]=this.elem[k];
        k=k+1;
    }
    this.cant--;
}
```

5. L1.eliminarPrim() : Método que elimina el elemento de la primer posición.

```
public void eliminarPrim(){
    eliminarIesimo(0);
}
```

6. L1.eliminarUlt() : Método que elimina el último elemento de la lista L1.

```
public void eliminarUlt(){
    eliminarIesimo(this.cant);
}
```

8. L1.eliminarTodo(x) : Método que elimina todos los elementos x de la lista L1.

```
public void eliminarTodo(int x){
    for (int i=0;i<cant;i++){
        if(elem[i]==x){
            eliminarIesimo(i);
        }
    }
}
```

8. L1.eliminarPares() : Método que elimina los elementos pares de la lista L1. Verificar en listas dónde todos los elementos sean pares.

```
public void eliminarPares(){
    for (int i=0;i<cant;i++){
        if((elem[i] % 2)==0){
            eliminarIesimo(i);
        }
    }
}
```

9. L1.eliminarUnicos() : Método que elimina los elementos que aparecen solo una vez en la lista L1.

```
public void eliminarUnicos(){
    for (int i=0;i<cant;i++){
        if(!repetido(elem[i])){
            eliminarIesimo(i);
        }
    }
}
```

10. L1.eliminarTodo(L2) : Método que elimina todos los elementos de la lista L1, que aparecen en la lista L2.

```
private boolean repetido (int x){
    int c=0;
    for (int i=0;i<cant;i++){
        if(elem[i]==x) c++;
    }
    return c>1;
}
```

11. L1.pasarDigitos(n) : Método que pasa los dígitos del entero n, a la Lista L1. Ejemplo: Si n = 546781, L1 = []. El resultado es L1 = [5, 4, 6, 7, 8, 1]

```
public void pasarDigitos(int x){
    while(x!=0){
        int num = x %10;
        x=x/10;
        insertarPrim(num);
```

```
        }
    }
```

12. L1.rotarIzqDer(n) : Método que hace rotar los elementos de la lista L1 n-veces de izquierda a derecha. Sugerencia, utilizar los métodos de insertar y eleminar por los extremos de la Lista.

```
public void rotarIzqDer(int n){
    while(n>0){
        int valor = getEle(cant-1);
        insertarPrim(valor);
        eliminarUlt();
        n--;
    }
}
```

13. L1.rotarDerIzq(n) : Método que hace rotar los elementos de la lista L1 n-veces de derecha a izquierda.

```
public void rotarDerIzq(int n){
    for (int i = 0; i < n; i++) {
        int x = getEle(0);
        eliminarPrim();
        insertarUlt(x);
    }
}
```

14. L1.eliminarPrim(n) : Método que eliminar los primeros n-elementos de la lista L1.

```
public void eliminarPrim(int n){
    for (int i = 0; i < n; i++) {
        eliminarPrim();
    }
}
```

15. L1.eliminarUlt(n) : Método que elimina los n-últimos elementos de la lista L1.

```
public void eliminarUlt(int n){
    for (int i = 0; i < n; i++) {
        eliminarUlt();
    }
}
```

16. L1.insertarIesimo(L2, i) : Método que insertar los elementos de la lista L2 en la lista L1, desde la posición i.

```
public void insertarIesimo(Lista L2, int pos){
    for (int i = 0; i < L2.cant; i++) {
        insertarIesimo(L2.getEle(i), pos+i);
    }
}
```

17. L1.insertarPrim(L2) : Método que insertar los elementos de la lista L2 al principio de la lista L1.

```
public void insertarPrim(Lista L2){
  for (int i = 0; i < L2.cant; i++) {
        insertarIesimo(L2.getEle(i), i);
    }
}
```

18. L1.insertarUlt(L2) : Método que insertar los elementos de la lista L2 al final de la lista L1.

```
public void insertarUlt(Lista L2){
        for (int i = 0; i < L2.cant; i++) {
            insertarUlt(L2.getEle(i));
        }
    }
```

19. L1.eliminarIesimo(i, n) : Método que elimina los n-elementos de la lista L1, desde la posición i.

```
public void eliminarIesimo(int i,int n){
 while (n!=0){

     eliminarIesimo(i);

     n--;

 }


}
```

20. L1.eliminarExtremos(n) : Método que eliminar la n-elementos de los extremos de la lista L1.

```
public void eliminarExtremos(int n){

    boolean b=true;

 while (n!=0 && cant!=0){

     if (b){

         b=false;

         eliminarPrim();

     }else{

         b=true;

         eliminarUlt();

     }

     n--;
```

```
    }
  }
```

21. L1.eliminarVeces(k) : Método que elimina los elementos que se repiten k-veces en la lista L1.

```
public void eliminarVeces(int k) {
      int i = 0;
      while (i < this.cant)
      {int count = 1;
        int j = i + 1;
              while (j < this.cant ) {

                  if(this.elem[i] == this.elem[j])
                          count++;
                  j++;
              }
              if (count == k) {
                     eliminarTodo(elem[i]);
              } else {
                  // Avanzar al siguiente elemento no repetido
                  i++;
              }
          }
      }
```

22. L1.eliminarAlternos() : Método que elimina los elementos de las posiciones alternas. (permanece, se elimina, permanece, se elimina, etc.)

```
  public void eliminarAlternos(){
      for (int i = 1; i < this.cant; i ++) {
         eliminarIesimo(i);
      }
  }
```

ANEXOS

```
public class Lista {
    private int elem[];
    private int cant;
    private int max;
    public Lista(int max){
        this.max = max;
        this.cant = 0;
        this.elem = new int[max];
    }
    public int getEle(int i){
        return elem[i];
    }
    public void insertarIesimo(int x, int i){
        int k = this.cant-1;
        while (k>=i){
            this.elem[k+1]=this.elem[k];
            k=k-1;
        }
        this.elem[i]=x;
        this.cant++;
    }

    public void insertarPrim(int x){
        insertarIesimo(x,0);
    }

    public void insertarUlt(int x){
        insertarIesimo(x, this.cant);
    }

    public void eliminarIesimo(int i){
        int k=i+1;
        while (k<this.cant){
            this.elem[k-1]=this.elem[k];
            k=k+1;
        }
        this.cant--;
    }

    public void eliminarPrim(){
        eliminarIesimo(0);
    }

    public void eliminarUlt(){
        eliminarIesimo(this.cant);
    }

    public void eliminarTodo(int x){
        for (int i=0;i<cant;i++){
            if(elem[i]==x){
                eliminarIesimo(i);
            }
        }
    }

    public void eliminarPares(){
        for (int i=0;i<cant;i++){
            if((elem[i] % 2)==0){
```

```
                eliminarIesimo(i);
            }
        }
    }

    public void eliminarUnicos(){
        for (int i=0;i<cant;i++){
            if(!repetido(elem[i])){
                eliminarIesimo(i);
            }
        }
    }

    private boolean repetido (int x){
        int c=0;
        for (int i=0;i<cant;i++){
            if(elem[i]==x) c++;
        }
        return c>1;
    }

    public void eliminarTodo(Lista L2){
        for (int i=0;i<this.cant;i++){
            if(contiene(L2,elem[i])){
                eliminarIesimo(i);
            }
        }
    }
    private boolean contiene(Lista L2,int x){
        for (int i=0;i<L2.cant;i++){
            if(L2.getEle(i)==x){
                return true;
            }
        }
        return false;
    }

    public void pasarDigitos(int x){
        while(x!=0){
            int num = x %10;
            x=x/10;
            insertarPrim(num);
        }
    }

    public void rotarIzqDer(int n){
        while(n>0){
            int valor = getEle(cant-1);
            insertarPrim(valor);
            eliminarUlt();
            n--;
        }
    }

    public void rotarDerIzq(int n){
        for (int i = 0; i < n; i++) {
            int x = getEle(0);
            eliminarPrim();
            insertarUlt(x);
        }
    }
```

```
    public void eliminarPrim(int n){
        for (int i = 0; i < n; i++) {
            eliminarPrim();
        }
    }

    public void eliminarUlt(int n){
        for (int i = 0; i < n; i++) {
            eliminarUlt();
        }
    }

    public void insertarIesimo(Lista L2, int pos){
        for (int i = 0; i < L2.cant; i++) {
            insertarIesimo(L2.getEle(i), pos+i);
        }
    }

    public void insertarPrim(Lista L2){
        for (int i = 0; i < L2.cant; i++) {
            insertarIesimo(L2.getEle(i), i);
        }
    }

    public void insertarUlt(Lista L2){
        for (int i = 0; i < L2.cant; i++) {
            insertarUlt(L2.getEle(i));
        }
    }

    public void eliminarIesimo(int i,int n){
        while (n!=0){
         eliminarIesimo(i);
         n--;
     }

    }

    public void eliminarExtremos(int n){
     boolean b=true;
     while (n!=0 && cant!=0){
         if (b){
             b=false;
             eliminarPrim();
         }else{
             b=true;
             eliminarUlt();
         }
         n--;
     }

    }

public void eliminarVeces(int k) {
    int i = 0;
    while (i < this.cant) {
        int count = 1;
        int j = i + 1;
        while (j < this.cant ) {
```

```
            if(this.elem[i] == this.elem[j])
                    count++;
            j++;
        }
        if (count == k) {
               eliminarTodo(elem[i]);
        } else {
            // Avanzar al siguiente elemento no repetido
            i++;
        }
    }
}


    public void eliminarAlternos(){
        for (int i = 1; i < this.cant; i ++) {
        eliminarIesimo(i);
    }

    }

    public String imprimir(){
        String e="";
        for (int i=0;i<cant;i++ ){
            e+="["+elem[i]+"]";
        }
        return e;
    }
```

COMENTARIOS

En primer lugar con esta tarea aprendimos mejor el uso de las listas así como distintos métodos curiosos que podemos hacer para un mejor manejo de ella y de este modo terminamos el trabajo avanzando más en nuestro dominio del tema.