Delgado Vasquez Erik No. Control 17211515

Instrucciones: Leer detalladamente el examen y realizar un programa en cualquier lenguaje de programación para resolver lo solicitado.

El conjunto de datos siguiente debe ser incluido en una lista doblemente enlazada.

1,2,3,4,5,6,7,8,9.

Realizar lo siguiente:

- Leer la lista de izquierda a derecha.
- Leer la lista de derecha a izquierda.
- Insertar: 10,11,13.
- Eliminar 8,1.
- Leer raíz.
- Leer final de la lista del lado derecho y lado izquierdo.
- Buscar: 0,7,8,1.

```
#Delgado Vasquez Erik No.Control 17211515

class Node():#Se crea una clase nodo para para crear los punteros next, prev y data.

   def __init__(self,data):

      self.data=data

      self.next=None

      self.prev=None

class Rorschach():#se crea una clase llamada Rorschach donde se pondran los metodos a utilizar

   def __init__(self):#Metodo donde se crearan los nodos y se ponen punteros para poder indicar determinadas posiciones en la lista

      self.root=None

      self.pixie=None

      self.brutus=None

      self.jeff=None

   def Null(self):#se crea el metodo Null para saber si la raiz tiene o no valores

      if self.root==None:

         return True

      else:

         return False
```

```
    def push(self,data):#se crea el metodo push, en este se ingresaran los datos
        if self.Null()==True:#si el metodo null devuelve valor true, significa que no hay valores en la raiz y por lo tanto se pueden meter datos en la misma
            self.root=self.pixie=Node(data)#se asigna el valor recibido a la raiz
            print("\nSe ha ingresado el elemento: " +str(data))#imprime el dato que se acaba de ingresar
        else:#si el valor devuelto del metodo null es false, se agregan los valores en los nodos siguientes
            monshiki=self.pixie#se usa una variable axiliar para almacenar el valor en un puntero
            self.pixie=monshiki.next=Node(data)#se asigna el valor al puntero
            self.pixie.prev=monshiki
            print("\nSe ha ingresado el elemento: " +str(data))
    def ChaoNumb(self,mushu):#se crea el metodo ChaoNumb para eliminar los datos que se pide borrar
        self.pixie=self.root#se iguala el puntero pixie al de la raiz
        monshiki=1#se utiliza una variable auxiliar y se le asigna el valor de 1
        if self.Null()==True:#si el valor devuelto por el metodo null es true significa que no hay ningun dato en la lista
            print("\nLa lista esta vacia.")
        else:
            while self.pixie.data!=mushu and monshiki==1:#siempre y cuando el elemento de la raiz sea diferente al valor recibido se realizara lo siguiente
                if self.pixie.next!=self.root:#si el nodo siguiente a la raiz es diferente a la raiz sehace lo siguiente
                    self.brutus=self.pixie#se asigna el valor de pixie a brutus
                    self.pixie=self.pixie.next#se asigna el valor de pixie.next a pixie
                else:
                    monshiki=0
            if monshiki==0: #si el valor de la variable auxiliar es cero, se dice que el dato no existe
                print("\nNo existe el dato a eliminar.")
            else:
                if self.root == self.pixie:#si la raiz es igual al nodo pixie, se asigna el valor de la raiz al nodo siguiente de pixie y se elimina el dato
                    self.root=self.pixie.next
```

```
            print("\nSe ha eliminado el elemento: " +str(mushu))
        else:
            self.brutus.next=self.pixie.next
            self.pixie=None
            print("\nSe ha eliminado el elemento: " +str(mushu))
    def ReadRoot(self):#metodo apra leer el elemento de la raiz
        if self.Null()==True:#si el valor devuelto por elmetodo null es true, se dice que no existe ninguna raiz
            print("\nNo existe una raiz.")
        else:
            print("\nLa raiz es: " +str(self.root.data))#en caso contrario, imprime el elemento que contenga la raiz
    def Lpeek(self):#metodo para leero la lista de izquierda a derecha
        mushu=self.root#se asigna la raiz a una variale
        if self.Null()==True:#si el valor devuelto por el metodo null es true se dice que existe ningun elemento en la lista
            print("\nNo hay datos en la lista.")
        else:#en caso contrario se imprime cada elemento que exista en la lista de izquierda a derecha
            while mushu:
                print("["+str(mushu.data)+"]")
                mushu=mushu.next#se asigna el valor de la variable auxiliar a el siguiente nodo para que prosiga con la cadena de impresion
    def Rpeek(self):#metodo para leer la lista de derecha a izquierda
        mushu=self.pixie#se asigna el valor del nodo a la variable
        if self.Null()==True:#si el valor devuelto por el metodo null es true se dice que existe ningun elemento en la lista
            print("\nNo hay datos en la lista.")
        else:#en caso contrario se imprime cada elemento que exista en la lista de derecha a izquierda
            while mushu:
                print("["+str(mushu.data)+"]")
                mushu=mushu.prev#se asigna el valor de la variable auxiliar a el nodo anterior para que prosiga con la cadena de impresion
```

```
    def LastData(self):#metodo en el que se lee el elemento del ultimo nodo
        if self.Null()==True:#si el valor devuelto por el metodo null es true se dice que existe ningun elemento en la lista
            print("\nNo hay datos en la lista.")
        else:#en caso contrario se asignan los valores del puntero siguiente a brutus al puntero pixie
            self.pixie=self.brutus.next#luego el valor siguiente al nuevo puntero pixie al puntero jeff
            self.jeff=self.pixie.next#por ultimo el siguiente a jeff, se asigna a jeff
            self.jeff=self.jeff.next
            print("\nEl ultimo elemento de la lista es: " +str(self.jeff.next.data))#imprime el elemento guardado en el ultimo nodo
    def FirstData(self):#metodo para leer el elemento del primer nodo
        if self.Null()==True:#si el valor devuelto por el metodo null es true se dice que existe ningun elemento en la lista
            print("\nNo hay datos en la lista.")
        else:#en caso contrario se imprime el valor que se ecuentra en la raiz, el cual es el primer nodo
            print("\nEl primer elemento es: " +str(self.root.data))
    def Search(self, archie):#metodo para buscar los elementos en la lista
        self.pixie = self.root#se asigna la raiz a pixie
        monshiki = 1#Variable auxiliar inicializada en 1
        if self.Null() == True:#si el valor devuelto por el metodo null es true se dice que existe ningun elemento en la lista
                print("\nLa lista esta vacia.")
        else:#mientras el elemento de la raiz sea diferente del elementoa buscar, y monshiki sea 1, se ejecuta lo siguiente
            while self.pixie.data != archie and monshiki ==1:  #si el valor siguiente a la raiz es diferente a la raiz
                if self.pixie.next != self.root:  #se asigna el valor de la raiz al siguiente nodo
                    self.pixie = self.pixie.next
                else:
                    monshiki = 0#se iguala a cero monshiki
                if monshiki == 0:#si monshiki es igual a 0 entinces el dato si existe en la lsita
                    print("\nEl dato " +str(archie)+" existe en la lista.")
```

```
            return self.pixie
        else:#en caso contrario, el dato no existe
            print("\nNo existe el dato " +str(archie)+" en la lista.")
            return self.pixie
    def Searcho(self, archie):#metodo para buscar los elementos en la lista
        monshiki = self.root#se asigna el calor de la raiz a la variable auxiliar
        keeper = False#se usa otra variable auxiliar y se dice que devuelve tipo de valor false
        while monshiki:#mientras la raiz; si el dato de esta, es igual al elemento a buscar; el valor de devuelto de keeper cambia a true;
            if monshiki.data ==archie:
                keeper = True
                print("\nEl elemento " + str(archie) + " pertenece a un nodo y existe en la lista.")#y entonces se dice que el elemento exiaste en la lista
                return keeper
            else:#en caso contrario, la raiz se iguala a su nodo siguiente; y si monshiki es igual a la raiz, keeper devuelve valor de false, por lo tanto;
                monshiki = monshiki.next
                if monshiki == self.root:
                    keeper = False
                    print("\nEl elemento " +str(archie)+ " no se encuentra en ningun nodo.")#el elemento no existe en la lista
                    return keeper
class Do():#clase para mandar a llamar a los metodos
    def DoIt(self):#metodo donde se manda a llamar los metodos
        mishi=Rorschach()#objeo de la clase Rorschach para mandar a llamar a los metodos
        mishi.push(1)
        mishi.push(2)
        mishi.push(3)
        mishi.push(4)
        mishi.push(5)           #se madnan a llamar los metodos con las instrucciones indicadas anteriormente.
        mishi.push(6)
        mishi.push(7)
```

```
        mishi.push(8)
        mishi.push(9)
        print("\nLista de izquierda a derecha.")
        mishi.Lpeek()
        input()
        print("\nLista de derecha a izquierda.")
        mishi.Rpeek()
        input()
        mishi.push(10)
        mishi.push(11)
        mishi.push(13)
        mishi.ChaoNumb(8)
        mishi.ChaoNumb(1)
        mishi.ReadRoot()
        mishi.LastData()
        mishi.FirstData()
        mishi.Search(0)
        mishi.Searcho(7)
        mishi.Search(8)
        mishi.Search(1)
        input()
Go=Do()
Go.DoIt()
```

Se ingresan los elementos 1, 2, 3, 4, 5, 6, 7, 8 y 9.

```
Se ha ingresado el elemento: 1

Se ha ingresado el elemento: 2

Se ha ingresado el elemento: 3

Se ha ingresado el elemento: 4

Se ha ingresado el elemento: 5

Se ha ingresado el elemento: 6

Se ha ingresado el elemento: 7

Se ha ingresado el elemento: 8

Se ha ingresado el elemento: 9
```

Se leen los elementos de la lista de derecha a izquierda.

```
Lista de izquierda a derecha.
[1]
[2]
[3]
[4]
[5]
[6]
[7]
[8]
[9]
```

Se leen los elementos de la lista de derecha a izquierda.

```
Lista de derecha a izquierda.
[9]
[8]
[7]
[6]
[5]
[4]
[3]
[2]
[1]
```

Se ingresan los elementos 10, 11 y 13.

```
Se ha ingresado el elemento: 10

Se ha ingresado el elemento: 11

Se ha ingresado el elemento: 13
```

Se eliminan los elementos 8 y 1.

```
Se ha eliminado el elemento: 8

Se ha eliminado el elemento: 1
```

Se lee el elemento guardado en la raíz.

```
La raiz es: 2
```

Se leen los elementos almacenados en el ultimo y primer nodo respectivamente.

```
El ultimo elemento de la lista es: 13

El primer elemento es: 2
```

Se buscan los elementos 0, 7, 8 y 1; y se muestra si se encuentran o no en la lista.

```
No existe el dato 0 en la lista.

El elemento 7 pertenece a un nodo y existe en la lista.

No existe el dato 8 en la lista.

No existe el dato 1 en la lista.
```