**BURSA ULUDAĞ ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ BİLGİSAYAR MÜHENDİSLİĞİ**

**2022 – 2023 EĞİTİM YILI BAHAR DÖNEMİ**

**VERİ YAPILARI DERSİ**

**PALİNDROME ÖDEVİ**

**Ad Soyad : Mert Gökmen**

**Numara : 032190004**

**Projede verilen tasarım koşulları : Bağlı liste kullanarak radix sort yapmak.**

**Kullanılan veri modeli : Düğümler ve düğümlerden oluşan bağlı liste.**

**Varsayımlar : Dizideki sayıların 2 basamaklı olması.**

**Örnek Çıktılar :**

**metin, ekran, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Kod :**

using System;

using System.Collections;

namespace Sandman

{

class Program

{

class node

{

public int veri;

public node ileri;

public node geri;

public node(int x)

{

veri = x;

ileri = null;

geri = null;

}

}

class queue

{

public node bas;

public node son;

public queue()

{

this.bas = null;

this.son = null;

}

//Kuyruğa sırasuz bir şekilde elemanları ekler.

public void enqueue(node node)

{

if (bas == null)

{

this.bas = node;

this.son = node;

}

else

{

node.geri = this.son;

this.son.ileri = node;

this.son = node;

}

}

//Kuyruktan eleman siler.

public int dequeue()

{

if (this.size() > 0)

{

int mesi = this.bas.veri;

this.bas = this.bas.ileri;

this.bas.geri = null;

return mesi;

}

else

{

Console.WriteLine("Listede silmek için eleman yok.");

return -1;

}

}

//Kuyruğun en sonundaki düğümün verisini geri döner.

public int peek()

{

if (this.size() > 0)

{

return this.son.veri;

}

else

{

Console.WriteLine("Listede eleman yok.");

return -1;

}

}

//Kuyruğun boş olup olmadığını döndürür.

public bool isEmpty()

{

return this.bas == null;

}

//Kuyruğu gösterir.

public void printQueue()

{

node temp = this.bas;

while (temp != null)

{

Console.Write(temp.veri + " <--> ");

temp = temp.ileri;

}

}

//Kuyrukta kaç düğüm olduğunu döndürür.

public int size()

{

int count = 0;

node temp = this.bas;

while (temp != null)

{

count++;

temp = temp.ileri;

}

return count;

}

//Kuyruğun başına düğüm ekler.

public void addtohead(node node)

{

if (bas == null)

{

this.bas = node;

this.son = node;

}

else

{

node.ileri = this.bas;

this.bas.geri = node;

this.bas = node;

}

}

//Kuyruğun sonuna düğüm ekler.

public void addtoend(node node)

{

if (bas == null)

{

this.bas = node;

this.son = node;

}

else

{

node.geri = this.son;

this.son.ileri = node;

this.son = node;

}

}

}

class nodenumber

{

public int veri;

public nodenumber ileri;

public nodenumber geri;

public queue ones;

public queue tens;

public nodenumber(int x)

{

this.veri = x;

this.ileri = null;

this.geri = null;

this.ones = new queue();

this.tens = new queue();

}

}

class queuenumber

{

public nodenumber bas;

public nodenumber son;

public queuenumber()

{

this.bas = null;

this.son = null;

}

//Kuyruğa sırasuz bir şekilde elemanları ekler.

public void enqueue(nodenumber nodenumber)

{

if (bas == null)

{

this.bas = nodenumber;

this.son = nodenumber;

}

else

{

nodenumber.geri = this.son;

this.son.ileri = nodenumber;

this.son = nodenumber;

}

}

//Kuyruktan eleman siler.

public int dequeue()

{

if (this.size() > 0)

{

int mesi = this.bas.veri;

this.bas = this.bas.ileri;

this.bas.geri = null;

return mesi;

}

else

{

Console.WriteLine("Listede silmek için eleman yok.");

return -1;

}

}

//Kuyruğun en sonundaki düğümün verisini geri döner.

public int peek()

{

if (this.size() > 0)

{

return this.son.veri;

}

else

{

Console.WriteLine("Listede eleman yok.");

return -1;

}

}

//Kuyruğun boş olup olmadığını döndürür.

public bool isEmpty()

{

return this.bas == null;

}

//Kuyruğu gösterir.

public void printQueue()

{

nodenumber temp = this.bas;

while (temp != null)

{

Console.Write(temp.veri + " <--> ");

temp = temp.ileri;

}

Console.WriteLine("NULL");

}

//Kuyrukta kaç düğüm olduğunu döndürür.

public int size()

{

int count = 0;

nodenumber temp = this.bas;

while (temp != null)

{

count++;

temp = temp.ileri;

}

return count;

}

//Kuyruğun başına düğüm ekler.

public void addtohead(nodenumber node)

{

if (bas == null)

{

this.bas = node;

this.son = node;

}

else

{

node.ileri = this.bas;

this.bas.geri = node;

this.bas = node;

}

}

//Kuyruğun sonuna düğüm ekler.

public void addtoend(nodenumber node)

{

if (bas == null)

{

this.bas = node;

this.son = node;

}

else

{

node.geri = this.son;

this.son.ileri = node;

this.son = node;

}

}

}

static void InsertOnes(int[] array,queuenumber q)

{

for (int i = 0; i < array.Length; i++)

{

int digit = array[i] % 10;

nodenumber temp = q.bas;

while (digit != temp.veri)

{

temp = temp.ileri;

}

temp.ones.enqueue(new node(array[i]));

}

}

static void InsertTens(int[] array, queuenumber q)

{

for (int i = 0; i < array.Length; i++)

{

int digit = array[i] / 10;

nodenumber temp = q.bas;

while (digit != temp.veri)

{

temp = temp.ileri;

}

temp.tens.enqueue(new node(array[i]));

}

}

static void SortByTens(queuenumber q)

{

nodenumber n = q.bas;

while (n != null) {

n.tens.printQueue();

n = n.ileri;

}

}

static void SortByOnes(queuenumber q)

{

nodenumber n = q.bas;

while (n != null)

{

n.ones.printQueue();

n = n.ileri;

}

}

static void Main(string[] args)

{

queuenumber q = new queuenumber();

int[] iarray = { 41, 46, 13, 53, 29, 64 };

q.enqueue(new nodenumber(0));

q.enqueue(new nodenumber(1));

q.enqueue(new nodenumber(2));

q.enqueue(new nodenumber(3));

q.enqueue(new nodenumber(4));

q.enqueue(new nodenumber(5));

q.enqueue(new nodenumber(6));

q.enqueue(new nodenumber(7));

q.enqueue(new nodenumber(8));

q.enqueue(new nodenumber(9));

//Dizideki sayıları kuyruğa onlar basamağına göre yerliştir.

InsertTens(iarray,q);

Console.Write("Sort by tens : ");

//Sıralanan sayıları bastırır.

SortByTens(q);

//Dizideki sayıları kuyruğa birler basamağına göre yerliştir.

InsertOnes(iarray, q);

Console.Write("\nSort by ones : ");

//Sıralanan sayıları bastırır.

SortByOnes(q);

}

}

}