**BURSA ULUDAĞ ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ BİLGİSAYAR MÜHENDİSLİĞİ**

**2022 – 2023 EĞİTİM YILI BAHAR DÖNEMİ**

**VERİ YAPILARI DERSİ**

**ÖNCELİKLİ KUYRUK ÖDEVİ**

**Ad Soyad : Mert Gökmen**

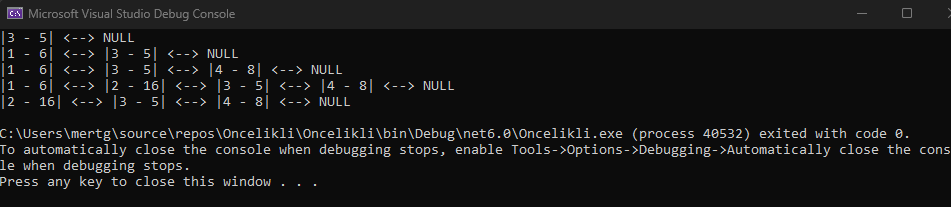
**Numara : 032190004**

**Projede verilen tasarım koşulları : İçerisinde öncelik ve veri olan düğümleri olan kuyruk yapısı kullanarak önceliklere göre kuyruğu sıralamak.**

**Kullanılan veri modeli : İçerisinde öncelik ve veri olan düğümleri olan kuyruk.**

**Varsayımlar : Kuyruğa elemanlar eklenirken verilerin int veri tipinde girilmesi.**

**Örnek Çıktılar :**

****

**Kod :**

using System;

namespace Sandman

{

class Program

{

class node

{

public int veri;

public int oncelik;

public node ileri;

public node geri;

public node(int x)

{

veri = x;

oncelik = 0;

ileri = null;

geri = null;

}

public node(int x,int once)

{

veri = x;

oncelik = once;

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;

}

}

//Kuyruğa sıralı bir şekilde elemanları ekler.

//Elemanları node classının öncelik propuna göre sıralar.

public void enqueueSorted(node node) {

if (this.bas == null)

{

this.bas = node;

this.son = node;

}

else if (this.size() == 1)

{

if (this.bas.oncelik > node.oncelik)

{

addtohead(node);

}

else

{

addtoend(node);

}

}

else

{

node temp= this.bas;

while (temp.ileri!=null && temp.ileri.oncelik <= node.oncelik)

{

temp = temp.ileri;

}

node.ileri = temp.ileri;

temp.ileri = 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.oncelik+ " - " + temp.veri+"|" + " <--> ");

temp = temp.ileri;

}

Console.WriteLine("NULL");

}

//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;

}

}

}

static void Main(string[] args)

{

queue q = new queue();

q.enqueueSorted(new node(5,3));

q.printQueue();

q.enqueueSorted(new node(6, 1));

q.printQueue();

q.enqueueSorted(new node(8, 4));

q.printQueue();

q.enqueueSorted(new node(16, 2));

q.printQueue();

q.dequeue();

q.printQueue();

}

}

}