**שיעורי בית יסודות מטלת תור – אופיר הופמן י3**

**מחלקת QueueA**

class QueueA<T>

{

private T[] arr;

private int head;

private int tail;

private int cnt;

public QueueA(int n)

{

if (n > 0)

{

this.arr = new T[n];

this.head = 0;

this.tail = 0;

this.cnt = 0;

}

}

public bool Insert(T val)

{

if (this.cnt == this.arr.Length)

return false;

this.arr[(tail) % this.arr.Length] = val;

this.cnt++;

this.tail++;

return true;

}

public T Remove()

{

if (cnt == 0)

return arr[-1];

T save = this.arr[head % this.arr.Length];

this.arr[head % this.arr.Length] = default(T); // reset arr cell value

head++;

cnt--;

return save;

}

public bool IsEmpty()

{

return this.cnt == 0;

}

public override string ToString()

{

string s = "";

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

{

s += this.arr[i]+",";

}

return s;

}

}

**פעולות סטאטיות Copy וReverse**

public static void QueueCopy(QueueA<int> queue1, QueueA<int> queue2, int n)

{

for (int i = 0; i < n; i++)

{

int temp = queue1.Remove();

queue1.Insert(temp);

queue2.Insert(temp);

}

}

public static void QueueReverse(QueueA<int> queue1, QueueA<int> queue2, int n)

{

// Make A new temporary array with same values as queue1

int[] arr = new int[n];

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

{

int temp = queue1.Remove();

queue1.Insert(temp);

arr[i] = temp;

}

// Insert() every value from arr in queue2 starting from the end

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

{

queue2.Insert(arr[arr.Length-i-1]);

}

}

**4. פעולת** whatDoIDo **ממיינת באופן רקורסיבי את התור שמתקבל כפרמטר.**

**לאחר הרצה של הפעולה על תור בגודל 10 גיליתי שהתשובה שלי הייתה נכונה.**

**5. מחלקת PriorityQueue**

**תת מחלקה – Value**

// Each object holds a numeric value and a "priority"

class Value

{

private int num;

private int priority;

public Value(int num, int priority)

{

this.num = num;

this.priority = priority;

}

public int GetNum()

{

return this.num;

}

public int GetPriority()

{

return this.priority;

}

public override string ToString()

{

return this.num + ", P=" + this.priority;

}

}

class PriorityQueue

{

private Value num;

private int TopPriority;

private QueueA<Value> pq; // כל עצם במחלקה הוא תור בעל איברים מטיפוס Value

private int size;

public PriorityQueue(int size)

{

this.size = size;

this.pq = new QueueA<Value>(size);

this.TopPriority = 5;

}

public bool PqInsert(int x, int priority)

{

if (priority > 0)

{

this.num = new Value(x, priority);

this.pq.Insert(num);

return true;

}

return false;

}

public bool PqIsEmpty()

{

return this.pq.IsEmpty();

}

public int PqRemove()

{

if (this.pq.IsEmpty())

return this.pq.Remove().GetNum();

int maxPriorityValue = 0;

int maxPriorityIndex = 0;

Value temp;

for (int i = 0; i < this.size; i++)

{

temp = this.pq.Remove();

if (temp.GetPriority() > maxPriorityValue)

{

maxPriorityValue = temp.GetPriority();

maxPriorityIndex = i;

}

this.pq.Insert(temp);

}

bool cont = true;

while (cont)

{

temp = this.pq.Remove();

if (maxPriorityIndex > 0)

this.pq.Insert(temp);

else

cont = false;

maxPriorityIndex--;

}

return maxPriorityValue;

}

public override string ToString()

{

return this.pq.ToString();

}

}

**תור וחיבור קצוות**

לאחר מעבר על 130 מספרים מן הרשימה והוספה/הוצאה מהתור על פי החוקים, סכום קצוות התור הוא 100. פתרון:

**פעולת SumEdges**

static public int SumEdges(QueueA<int> q, int n)

// n equals to number of items in the queue

{

QueueA<int> newQ = new QueueA<int>(30); // Temp Queue

// Copy q to newQ

QueueCopy(q, newQ, n);

// Find first value in q

int first = newQ.Remove();

int last = 0;

if (newQ.IsEmpty())

last = first; // if there was only 1 item in Queue

else

{

// Find last value in q

for (int i = 0; i < n - 1; i++)

{

last = newQ.Remove();

}

}

return first + last;

}

**Main:**

// Read file into array of integers

int MaxLength = 30;

string s = "";

using (StreamReader sr = new StreamReader("a.txt"))

{

s = sr.ReadLine();

}

string[] numbersStrings = s.Split(' ');

int[] numbers = new int[numbersStrings.Length];

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

{

numbers[i] = int.Parse(numbersStrings[i]);

}

QueueA<int> q1 = new QueueA<int>(MaxLength); // New Queue

int saveIndex = 0; // saves the index of number in list which after adding // - sum of edges == 100

bool cont = true;

int cnt = 0; // counts the numbers of items in the queue

// Go over numbers[]

for (int i = 0; i < numbers.Length && cont; i++)

{

// for each number in numbers that is >0

if (numbers[i] > 0)

{

// Add to queue

q1.Insert(numbers[i]);

// Check If sum of edges == 100

if (SumEdges(q1, cnt + 1) == 100)

{

saveIndex = i; // save index

cont = false; // stop loop

}

cnt++;

}

// If item==0 remove from head of queue

else if (numbers[i] == 0)

{

q1.Remove();

cnt--;

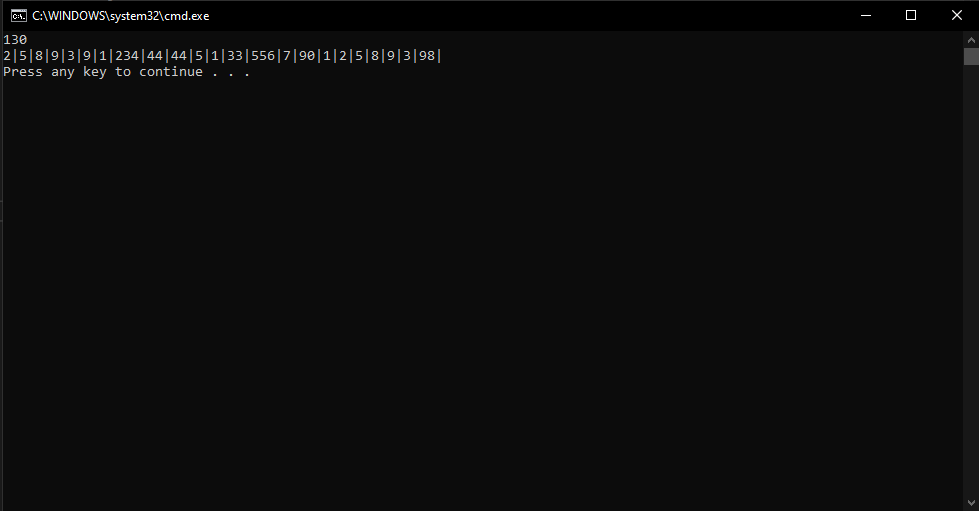
}

}

Console.WriteLine(saveIndex);

Console.WriteLine(q1);

**צילום מסך של הרצה למטה**

****

**למעלה - מספר המספרים מן הרשימה שעליהם עברה התוכנית**

**מתחת – התור לאחר 130 מעברים על מספרים מהרשימה והוספה במידה והמספר מעל 0 והוצאה במידה והוא 0.**