מבוא למדעי המחשב – כתה יא-1

תאריך: 22.1.2026

מבוא לתורים <Queue<T> - דף הסבר + תירגול #1

חלק א': מבוא לתורים

חלק זה הוא תקציר (ולא תחליף) להסברים בכיתה ובספר הלימוד.

1) מה זה תור? כפי שאנחנו מכירים בחיים: משהו שצובר ערכים. הראשון שנכנס הוא הראשון שיוצא. FIFO = First In First Out
2) איך אנחנו ממשים תור? ← אנחנו לא מממשים (לא כותבים את הקוד) למחלקה Queue<T> !! אנחנו רק משתמשים בה.
3) המחלקה היא גנרית – ולכן התור יכול להיות של טיפוס כלשהו – ע"י השימוש ב <T> כפי שכבר הכרנו. בשביל הפשטות, חלק גדול מהתרגילים יהיו על Queue<int>/
4) ממשק – זה הפעולות שהמחלקה (כל מחלקה) מספקת לנו. המחלקה Queue<T> מספקת את הפעולות הבאות:

```
Queue<T>();  // constructor.  Creates an empty queue
             //  e.g. Queue<int> myQ = new Queue<int>();
void Insert(T item);
T Remove();
T Head();  // same as Remove, but the item remains in the queue
bool IsEmpty()
```

5) הפעולות האלו נמצאות בתוך המחלקה. ולכן אנחנו ניגשים אליהן "כרגיל" – דרך האובייקט של המחלקה. לדוגמא:

```
myQ.insert(10);
int val;
if (! myQ.IsEmtpy())
  val = myQ.remove();
```

6) אסור לבצע Remove or Head על תור מבלי לוודא קודם שהתור איננו ריק ! כלומר חובה לבדוק IsEmpty() לפני הפעולה.
7) לפני שנשתמש בתור נייצר אותו על ידי הקריאה לבנאי (ללא ארגומנטים) שיוצר תור ריק.

סיכום הממשק:

בדף הבא נמצא סיכום של הממשק המחייב לבגרות לתור הגנרי. זה לקוח מתוך האתר של הילן קדמן.

ממשקים למבני הנתונים (חומר מחייב לבגרות)

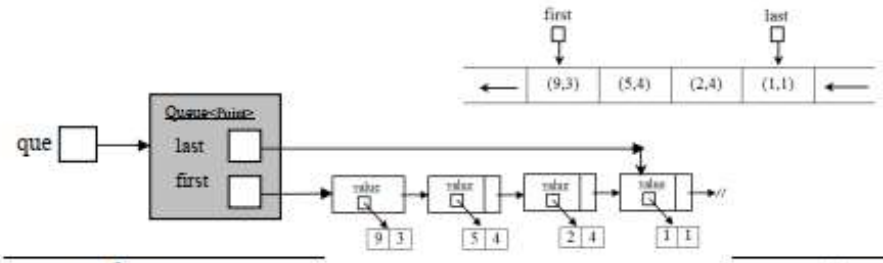
תור גנרי Queue<T>

תיאור הפעולה	חתימת הפעולה	סיבוכיות
בנאי:		
פעולה הבונה תור ריק	Queue ()	O(1)
שאילתות:		
תור-ריק? פעולה המחזירה אמת אם התור ריק ושקר אחרת	**bool** IsEmpty ()	O(1)
ראש-התור () הפעולה מחזירה את האיבר שבראש התור מבלי להוציאו. תנאי קדם: התור לא ריק	T Head ()	O(1)
פעולה המחזירה מחרוזת המתארת את מצב התור באופן הבא: $[x_1, x_2, x_3, \ldots, x_n]$ כך ש- x_1 בראש התור ו- x_n בסופו. (x_n איבר חדש יוכנס אחרי) (*) סיבוכיות: n מייצג את מספר האיברים בתור אם T עצם מטיפוס פשוט ← O(n) ואם T מייצג אוסף באורך כלשהו ← $O(n \cdot \lvert T \rvert)$ $\lvert T \rvert$ אורך T (למשל: תור הדפסות: T הוא עצם מסוג מסמך להדפסה שיש בו m דפים)	**string** ToString ()	O(n·|T|) (*)
פקודות:		
הכנס-לתור (x) הפעולה מוסיפה את x לסוף התור	**void** Insert (T x)	O(1)
הוצא-מהתור () הפעולה מוציאה ומחזירה את האיבר שבראש התור. תנאי קדם: התור לא ריק	T Remove ()	O(1)

משמעות **תנאי קדם**: הפעולה מניחה את תקינות הנתונים. התכנית/פעולה המזמנת את הפקודה חייבת לבדוק זאת. לדוגמה:

```
if (! q.IsEmpty () )
    x = q.Remove ();
```

אם התור-לא-ריק?
x ← הוצא-מהתור ()



חלק ב': תרגילי הדגמה ותירגול בכיתה

בחלק זה מופיעים תרגילים בסיסיים שמדגימים שימוש בתורים. לפתרון בזמן השיעור, והדגמה על ידי המורה.

הדגמה #1: פונקציה בשם `void Demo1()`. הפונקציה מייצרת תור ומכניסה אליו את האיברים `10, 20, 30`. לאחר מכן בלולאה, היא מוציאה וכותבת לפלט את האיברים שבתור.

הדגמה #2: פונקציה `void Demo2()` הפונקציה קולטת מספרים שלמים מהקלט עד שהיא קולטת מספר שלילי `-2 or -1`. כאשר היא קולטת `-1` היא כותבת לפלט את כל האיברים שבתור. כאשר היא קולטת `-2` היא מסתיימת (חזרה מהפונקציה).
הנחיה: משתמשים בפונקציה `void PrintAndEmptyQueue<T>(Queue<T q)` בכדי לכתוב לפלט את האיברים בתור.

הדגמה #3: פונקציה `void Demo3()`. הפונקציה קולטת מספרים שלמים ומכניסה אותם לתור. כאשר היא קולטת `-1` היא מדפיסה את התור עד כה, אך מבלי לרוקן אותו (איך זה אפשרי? ← שימוש בתור נוסף). כאשר היא קולטת `-2` היא מדפיסה את התור ומסיימת.
הנחיה: משתמשים בפונקציה `void PrintDoNotEmptyQueue <T>(Queue<T q)` בכדי לכתוב לפלט את האיברים בתור, מבלי לרוקן אותו.

הדגמה #4: פונקציה `void Demo4()`. הפונקציה קולטת מספרים שלמים ומכניסה אותם לתור עד שהיא קולטת `-1`. היא מחלקת את התור לשני תורים – אחד של מספרים זוגיים ואחד של מספרים אי-זוגיים. היא מדפיסה את כל אחד מהתורים עם הודעה מתאימה.

חלק ג' - תרגול עצמי מדורג -

לתירגול במעבדה או בבית, בכתב יד או מול מחשב, או (מומלץ) בהתייעצות נבונה עם AI

בחלק זה מופיעים תרגילים מהקל אל הכבד, עד לרמה שנדרשת לבגרות.

עליכם לעשות את כל התרגילים האלו.

המלצות כרגיל:

א. בצעו את התרגיל על דף נייר, תוך שימוש בציור.
ב. כתבו אלגוריתם לפי צעדים (1, 2, 3, וכו')
ג. היכן שרלבנטי ומקל – כתבו פונקציה שמבצעת תת-משימה
ד. כתבו במחשב באמצעות תכנה כמו notepad++. התייעצו עם AI לגבי השאלה והפתרון שלכם. השתמשו בפרומפט שבאתר בכדי לכוון אותו שאתם תלמידים, ושיעזור לכם בתהליך הלמידה ולא יפתור את התרגיל במקומכם.
ה. לבסוף העתיקו את הקוד לפרוייקט שלכם ב Visual Studio, עכשיו אתם יכולים להריץ עם קלטים שונים ולראות שהתרגילים השונים עובדים היטב.

הנחיות:

בתרגילים הבאים יש להשתמש **אך ורק** בפעולות המוגדרות למבנה תור:
`Insert, Remove, Head, IsEmpty.`
במידת הצורך מותר להשתמש **בתור עזר.**

תרגיל #1 **הדפסת תור וריקונו**

כתוב פונקציה שמקבלת תור ומדפיסה את כל איבריו לפי סדרם, תוך ריקון התור.

חתימה מוצעת:

```
static void PrintAndEmptyQueue<T>(Queue<T> q)
```

תרגיל #2 **הדפסת תור ללא ריקון**

כתוב פונקציה שמקבלת תור ומדפיסה את כל איבריו לפי סדרם **, מבלי לשנות את התור בסיום הפעולה.**

הנחיה: יש להשתמש בתור עזר.

חתימה מוצעת:

```
static void PrintDoNotEmptyQueue<T>(Queue<T> q)
```

תרגיל #3 **ספירת איברים בתור**

כתוב פונקציה שמחזירה את מספר האיברים בתור, מבלי לשנות את תוכן התור בסיום הפעולה.

חתימה מוצעת:

```
static int CountQueue<T>(Queue<T> q)
```

תרגיל #4 **שאלה 4 – סכום איברים בתור**

כתוב פונקציה שמקבלת תור של מספרים שלמים ומחזירה את סכום האיברים, מבלי לשנות את התור בסיום הפעולה.

חתימה מוצעת:

```
static int SumQueue(Queue<int> q)
```

הערת חשיבה: שימו לב לדמיון בין פתרונות תרגילים שונים בחלק זה.
למשל: מעבר על תור מבלי לשנות אותו בסיום הפעולה – פעם לצורך הדפסה, פעם לצורך חישוב סכום האיברים, ופעם למציאת הערך המקסימלי. למרות שהמטרה שונה, התבנית האלגוריתמית זהה.

תרגיל #5 מציאת מקסימום בתור

כתוב פונקציה שמקבלת תור של מספרים שלמים ומחזירה את הערך המקסימלי בתור, מבלי לשנות את התור בסיום הפעולה. **הניחו**: התור אינו ריק.

חתימה מוצעת:

```
static int MaxQueue(Queue<int> q)
```

תרגיל #6 הזזה מראש לסוף

כתוב פונקציה שמבצעת פעולה אחת של "הזזה" : האיבר שבראש התור יוצא ומוכנס לסוף התור. אם התור ריק – אין לעשות דבר.

חתימה מוצעת:

```
static void RotateOnce<T>(Queue<T> q)
```

תרגיל #7 הסרת הופעה ראשונה

כתוב פונקציה שמקבלת תור וערך נתון, ומסירה מהתור **את ההופעה הראשונה בלבד** של הערך (אם היא קיימת). אם הערך אינו קיים – התור נשאר ללא שינוי. יש לשמור על סדר שאר האיברים.

חתימה מוצעת:

```
static void RemoveFirstOccurrence<T>(Queue<T> q, T value)
```

תרגיל #8 סינון לפי תנאי

כתוב פונקציה שמקבלת תור של מספרים שלמים, ומשאירה בתור רק מספרים שמתחלקים ב־3, תוך שמירה על סדר האיברים.

חתימה מוצעת:

```
static void KeepOnlyMultiplesOf3(Queue<int> q)
```

תרגיל #9 סינון כפילויות מתור

נתון תור של מספרים שלמים. כתוב פונקציה שמסירה מהתור את כל הכפילויות, כך שבסיום הפעולה כל ערך יופיע **פעם אחת בלבד**, ותוך **שמירה על סדר ההופעה של האיברים בתור** שהפונקציה קבלה.
לדוגמה:
אם התור מכיל: [3 , 9 , 5 , 7 , 3 , 5 , 3]
אזי לאחר הפעולה התור יכיל: [9 , 7 , 5 , 3]

הנחיות:

- אין להשתמש במבני נתונים נוספים מלבד תורים
- מותר להשתמש בתור עזר

חתימה מוצעת:

```
static void RemoveDuplicates(Queue<int> q)
```

חלק ד' – תרגול תורים ברמת בגרות

בתרגילים הבאים יש להשתמש **אך ורק** בפעולות המוגדרות למבנה תור:
`Insert, Remove, Head, IsEmpty.`
מותר להשתמש בתור עזר, אלא אם נאמר אחרת.

תרגיל #10 (בגרות #1) בדיקת זהות בין שני תורים

נתונים שני תורים `q1` ו־`q2`.

שני תורים ייחשבו **זהים** אם:

- יש להם אותו מספר איברים
- אותם ערכים
- ובאותו סדר

כתוב פונקציה שמחזירה `true` אם התורים זהים, ו־`false` אחרת.

דרישה חשובה:
בסיום הפעולה, שני התורים חייבים להישאר **ללא שינוי**.

חתימה מוצעת:

```
static bool IsIdentical<T>(Queue<T> q1, Queue<T> q2)
```

תרגיל #11 (בגרות #2) בדיקת זהות באמצעות הזזות

נתונים שני תורים `q1` ו־`q2`.

כתוב פונקציה הבודקת האם ניתן להפוך את `q1` לזהה לתור `q2` באמצעות מספר כלשהו של פעולות הזזה מהראש לסוף התור (`RotateOnce`).

הנחיות:

- ניתן להיעזר בפונקציה `IsIdentical`
- מותר להשתמש בתור עזר
- בסיום הפעולה, שני התורים חייבים להישאר **ללא שינוי**

חתימה מוצעת:

```
static bool CanBecomeIdenticalByRotations<T>(Queue<T> q1, Queue<T> q2)
```

תרגיל #12 (בגרות #3) חיפוש זוג איברים שסכומם נתון

נתון תור `q` של מספרים שלמים וערך שלם `x`.

כתוב פונקציה שמחזירה `true` אם קיימים **שני איברים שונים בתור**, שסכומם שווה ל`x`, ו- `false` אחרת.

האיברים אינם חייבים להיות סמוכים בתור.

דרישות:

- אין לשנות את התור בסיום הפעולה
- אין להשתמש במבני נתונים נוספים מלבד תורים

חתימה מוצעת:

```
static bool TwoSum(Queue<int> q, int x)
```

תרגיל #13 **שאלה 4 – מיזוג שני תורים ממוינים**

נתונים שני תורים של מספרים שלמים:

- כל אחד מהם **ממויין בסדר עולה**

כתוב פונקציה שממזגת את שני התורים לתור חדש אחד,
כך שגם התור המוחזר יהיה ממויין בסדר עולה.

הנחיות:

- יש לשמור על סדר האיברים בכל אחד מהתורים המקוריים
- אין לשנות את התורים המקוריים בסיום הפעולה
- מותר להשתמש בתור עזר

חתימה מוצעת:

```
static Queue<int> MergeSortedQueues(Queue<int> q1, Queue<int> q2)
```

הוראות להרצת התרגילים עם Visual Studio.

הקבצים שאתם עובדים איתם

הקבצים הדרושים נמצאים ב GDRIVE. תוכלו להגיע אליהם בקלות דרך הקישור שנמצא באתר הבית - https://tzvimelamed.com/lab .

בתיקייה שאתם מקבלים על ה GDRIVE יש את הקבצים:

- `Program.cs` – מכיל את ה- `Main` ואת התפריטים. אין צורך לשנות את הקובץ הזה.
- `Node.cs` – נתון לכם. לא לשנות (וגם לא להשתמש בזה בתרגיל הזה).
- `Queue.cs` – נתון לכם. זאת מחלקת התור. זה הנושא של התרגיל. אל תשנו אותה.
- `Program_student.cs` – כאן עליכם לממש את התרגילים של חלק ג'.
- `Program_Bagrut_Level.cs` – כאן עליכם לממש את התרגילים של חלק ד'.

תזכורת - הנחיות:

א. יש להשתמש בציורים בשביל להמחיש את העבודה על התורים.
ב. תמיד לקחת בחשבון: תור יכול להיות ריק. כפי שנאמר בתחילה, אין לגשת או להסיר איבר מתור אם הוא ריק.