תכנות מתקדם – 150024

תרגיל בית מספר 2

מחלקות עם שטחים דינאמיים

<u>שים לב:</u>

א. הקפד על קְרִיאוּת התכנית ועל עימוד (Indentation).
ב. הקפד לבצע בדיוק את הנדרש בכל שאלה.
ג. בכל אחת מהשאלות יש להגדיר כל מחלקה ב 2 קבצים נפרדים. קובץ H וקובץ CPP.
ד. בכל אחת מהשאלות יש להגדיר פונקציות עזר במידת הצורך עבור קריאות התכנית.
ה. יש להגיש את התרגיל על פי ההנחיות להגשת תרגילים (המופיע באתר הקורס) וביניהם: השתמש בשמות משמעותיים עבור המשתנים.
יש לתעד את התכנית גם עבור פונקציות אותם הנך מגדיר/ה וכן על תנאים ולולאות וקטעי קוד מורכבים, ובנוסף, **דוגמת הרצה לכל תכנית בסוף הקובץ**!
.הגשה יחידנית - אין להגיש בזוגות.

הערה חשובה: לכל תרגיל בית מוגדר <u>שבוע אחד בלבד</u> להגשה, אלא אם כן קיבלת הוראה אחרת מהמרצה שלך. <u>תיבות ההגשה הפתוחות לא מהוות היתר להגשה באיחור.</u>

<u>שאלה מס' 1:</u>

א. הגדירו מחלקה **Point** לייצוג נקודה במישור.

ב. המחלקה תכלול את השדות **הפרטיים** באים:
- x – מספר עשרוני המייצג את המיקום על ציר x
- y – מספר עשרוני המייצג את המיקום על ציר y

ג. תחת הרשאה ציבורית, הוסף/י למחלקה **Point** לפחות את הבנאים הבאים:
- **default constructor** – בנאי שלא מקבל פרמטרים ומאתחל את השדות x ו-y באפס (כך שנגדיר נקודה בראשית הצירים)
- **constructor** – בנאי נוסף המקבל שני פרמטרים ומאתחל את שדות x ו-y לפי הפרמטרים שקבל
- **copy constructor** – בנאי העתקה שמקבל כפרמטר נקודה נוספת ומאתחל את ערכי ה-x וה-y של הנקודה הנוכחית בערכים של הנקודה שהתקבלה כפרמטר בהתאמה

ד. תחת הרשאה ציבורית, הוסיף/י למחלקה **Point** את המתודות הבאות:
- הצבה ואחזור (**get/set**) לכל אחד מהשדות
- **distance** - מתודה המקבלת כפרמטר נקודה קיימת, מחשבת ומחזירה את המרחק בין הנקודה הנוכחית לנקודה שהתקבלה כפרמטר. בשביל התרגיל, יש לשלוח את הפרמטר כ-cbr.

ה. הגדר/י מחלקה **Polygon** לייצוג מצולע כלשהו במישור.

ו. המחלקה **Polygon** תכלול את השדות **הפרטיים** הבאים:
- מצביע למערך של קודקודים בגודל לא ידוע (קודקוד הוא בעצם נקודה במישור – לפי מה שהגדרת בסעיף א)
- מספר הקודקודים במצולע (גודל מערך הקודקודים)

ז. תחת הרשאה ציבורית, הוסף/י למחלקה **Polygon** לפחות את הבנאים והמתודה ההורסת הבאים:
- **default constructor** – בנאי שלא מקבל פרמטרים ומאתחל את השדה של מספר הקודקודים ל- 0 ואת המצביע למערך הקודקודים ל nullptr.
- **constructor** – המקבל כפרמטר את מספר הקודקודים במצולע, מאתחל את שדה של מספר הקודקודים בהתאם וכן מאתחל את המצביע למערך הקודקודים ע"י הקצאה דינאמית של מערך קודקודים בגודל המתאים. הנקודות במערך יאותחלו ע"י קריאה באופן אוטומטי לבנאי ברירת המחדל שמאתחל את ערכי הנקודה באפסים.
- **copy constructor** – מקבל כפרמטר פוליגון קיים ומאתחל את הפוליגון הנוכחי באותם ערכים של הפוליגון שהתקבל כפרמטר. כלומר – אותו מספר קודקודים עם אותם ערכים. כמובן שיש להקצות מערך של קודקודים חדש ולהעתיק אליו את ערכי הנקודות.
- **destructor** – הורס. הורס במחלקה דינאמית צריך לזכור לשחרר את הזיכרון שהוקצה במחלקה.

ח. תחת הרשאה ציבורית, הוסף/י למחלקה **Polygon** לפחות את המתודות הבאות:
- מתודת אחזור (get) לכל שדה. עבור get של המערך חובה לבנות מערך חדש ולהחזיר את המערך החדש.
- (שאלה למחשבה – למה אין פה מתודות set?)
- **setPoint** - מתודה המקבלת פרמטר אחד מסוג Point ופרמטר שני שהוא אינדקס. יש לשלוח את הנקודה כ-cbv.
- על המתודה להציב את ערכי הקודקוד זה במערך הקודקודים באינדקס הנשלח כפרמטר. הקודקוד כבר קיים במערך עם ערכים אחרים או ערכים מאופסים והמתודה מחליפה את הערכים בערכים של הנקודה שהתקבלה כפרמטר.
- **perimeter** - מתודה המחשבת את היקף המצולע (סכום המרחקים בין כל שני קודקודים סמוכים). ניתן להניח שכל הנקודות מסודרות לפי הסדר בו הן מרכיבות את המצולע. (לא לשכוח את המרחק בין הנקודה הראשונה והאחרונה במערך!) על המתודה להחזיר מספר עשרוני.
- **isIdentical** - מתודה בוליאנית המקבלת כפרמטר מצולע ובודקת האם המצולע שהתקבל והמצולע הנוכחי זהים. בשביל התרגיל, יש לשלוח את הפרמטר כ-cbv.
- מצולעים יוגדרו זהים כאשר מספר הקודקודים שלהם שווה וערכי הקודקודים שלהם שווים. שימו לב, לא מחייב שהסדר של הקודקודים השמורים במערך יהיה זהה.

  לדוגמה: המצולע: (2,0) (1,1) (0,0) זהה למצולע: (0,0) (2,0) (1,1)

ט. לצורך הבנה עמוקה של הבנאים עליך להוסיף
a. במחלקה **Point** את ההדפסות הבאות:
- ב- default constructor יש להדפיס : in point default constructor
- ב- constructor שקיבל פרמטר אחד יש להדפיס : in point two parameter constructor
- ב- copy constructor יש להדפיס: in point copy constructor

b.
c. במחלקה **Polygon** את ההדפסות הבאות:
- ב- default constructor יש להדפיס : in poly default constructor
- ב- constructor שקיבל פרמטר אחד יש להדפיס : in poly one parameter constructor
- ב- copy constructor יש להדפיס: in poly copy constructor
- ב-destructor יש להדפיס : in destructor

י. כתוב תכנית ראשית הקולטת נתונים על שני מצולעים ואז מחשבת ומדפיסה את היקפם. יש לעבוד לפי הפירוט הבא:

a. **קלט כל אחד מהמצולעים**, יש להדפיס את ההודעה enter number of points: ואז תקלוט את מספר הקודקודים.
במקרה של קלט לא תקין התכנית תדפיס ERROR ותקלוט את המספר מחדש.

לאחר מכן יש להדפיס את ההודעה enter the point values: ואז לקלוט בלולאה את שיעורי הנקודות (הקודקודים).
הקלט יהיה מהצורה: (x1,y1) (x2,y2) … (xN,yN)

b. **חישוב היקף המעגלים מעוגל למספר השלם הקרוב ביותר**, (אפשר להשתמש בפונקציה round המוגדרת בספרייה cmath).

c. **הדפסת השטחים**: במידה והמצולעים זהים יש להדפיס equal ואז בשורה הבאה להדפיס perimeter: ואז את ההיקף (יש להדפיס את ההיקף רק פעם אחת)
במידה ואינם זהים יש להדפיס not equal ואז עבור כל מצולע בשורה נפרדת יש להדפיס perimeter: ואת היקפו

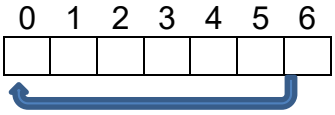
דוגמאות להרצת התכנית:

קליטת ריבוע שצלעותיו באורך 2 ולאחר מכן קליטת משולש בעל 3 צלעות שונות:	קליטת 2 משולשים ישרי זווית שצלעותיהם 3-4-5:
enter number of points: 4 in point default constructor in point default constructor in point default constructor in point default constructor in poly one parameter constructor enter the point values: (0,0) (0,2) (2,2) (2,0) in point two parameter constructor in point two parameter constructor in point two parameter constructor in point two parameter constructor enter number of points: 3 in point default constructor in point default constructor in point default constructor in poly one parameter constructor enter the point values: (1,1) (2,0) (3,1) in point two parameter constructor in point two parameter constructor in point two parameter constructor in point default constructor in point default constructor in point default constructor in poly copy constructor in destructor not equal perimeter:8 perimeter:5 in destructor in destructor	enter number of points: 3 in point default constructor in point default constructor in point default constructor in poly one parameter constructor enter the point values: (10,10) (10,14) (13,10) in point two parameter constructor in point two parameter constructor in point two parameter constructor enter number of points: 3 in point default constructor in point default constructor in point default constructor in poly one parameter constructor enter the point values: (13,10) (10,10) (10,14) in point two parameter constructor in point two parameter constructor in point two parameter constructor in point default constructor in point default constructor in point default constructor in poly copy constructor in destructor equal perimeter:12 in destructor in destructor

שאלה מס' 2:

מערך סדור הוא מערך שבו נשמרים הערכים לפי סדר הכנסתם למערך. כלומר ההכנסה היא תמיד למקום הבא הפנוי וההוצאה היא תמיד של האיבר שהוכנס ראשון למערך.

מערך מעגלי הוא מערך שיכול להכיל לכל היותר **capacity** ערכים בצורה מעגלית כלומר כאשר מגיעים לסוף המערך חוזרים להתחלה,
לדוגמא – עבור המערך הבא בגודל 7 האינדקס הבא אחרי אינדקס 6 הוא אינדקס 0.

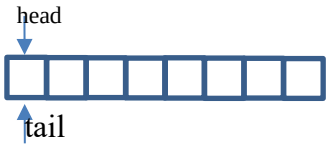


על מנת לנהל את המערך המעגלי נשמור בכל רגע נתון שני ערכים:
ראש המערך (head) - האינדקס של הערך הראשון שהוכנס למערך (האינדקס שממנו נוציא את הערך הבא)

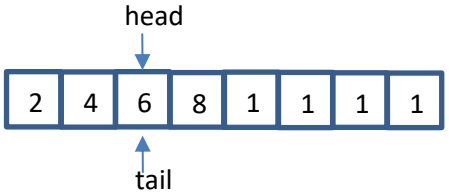
זנב המערך (tail) –האינדקס של המקום הפנוי הבא (האינדקס שאליו נכניס את הערך הבא להכנסה)

נשים לב: שכאשר head==tail המשמעות יכולה להיות אחת משתי אפשרויות:

א. או שהמערך ריק



ב. או שהמערך מלא



לכן
למרות שבפועל המערך המעגלי יכיל **לכל היותר capacity ערכים**, נגדיר את **גודלו הפיזי להיות capacity + 1**.
וכך:
כאשר הראש והזנב שווים המשמעות היא שהתור ריק
וכאשר head == (capacity+1) % (tail +1) המשמעות היא שהתור מלא.

א. הגדר מחלקה בשם **RoundVector** למימוש **מערך מעגלי** של מספרים שלמים.

השדות במחלקה יהיו:

- **capacity** – מספר האיברים המקסימלי האפשרי במערך. מספר הערכים הנמצאים בפועל במערך יהיה קטן שווה ל capacity.

- **vec** - מצביע למערך פיזי של שלמים בגודל של capacity+1. כמו שהסברנו למעלה.
- **head** – אינדקס של הערך הראשון במערך, אינדקס של הערך הבא בתור ליציאה מהמערך.
- **tail** – אינדקס של המקום הפנוי הבא במערך. לפניו נמצא האיבר האחרון במערך. האינדקס שאליו יכנס האיבר הבא למערך.

כל השדות יוגדרו בהרשאת גישה פרטית

ב. **בהרשאה ציבורית, הוסף למחלקה לפחות את הבנאים והמתודה ההורסת הבאים:**

- **constructor** - מקבל כפרמטר את גודל המערך. מאתחל בעזרתו את ה capacity כמבוקש, ומקצה את המערך vec באופן דינאמי על פי בגודל של capacity+1 ומאתחל את ערכי head ו-tail להיות 0.
- **copy constructor** – מקבל כפרמטר **RoundVector** קיים (קבוע שמועבר by reference &) ומאתחל את ה **RoundVector** הנוכחי באותם ערכים של המערך המעגלי שהתקבל כפרמטר.
  אך head ו tail -לא יועתקו אוטומטית אלא יש להעתיק את הערכים מהמערך שהתקבל כך שישבו בצורה סדרתית במערך החדש מאינדקס 0 והלאה. ולפי זה לעדכן אתhead ו tail -בהתאמה.
- **move constructor** - מקבל כפרמטר **RoundVector** קיים (שמועבר by reference &&) ומאתחל את ה **RoundVector** הנוכחי באותם ערכים של המערך המעגלי שהתקבל כפרמטר.
- **destructor** – הורס. הורס במחלקה דינאמית צריך לזכור לשחרר את הזיכרון שהוקצה במחלקה.

לצורך הבנה עמוקה של מתודות ה-ctor עליך להוסיף במחלקה RoundVector את ההדפסות הבאות:

- ב- constructor יש להדפיס : in constructor
- ב- copy constructor יש להדפיס: in copy constructor
- ב-destructor יש להדפיס : in destructor
- **אין צורך להדפיס כלום מתוך ה-move constructor**

ג. **תחת הרשאה ציבורית, הוסף למחלקה לפחות את המתודות הבאות:**

- **isEmpty** – מתודה בוליאנית שתחזיר אמת אם המערך לא מכיל ערכים. נבדוק זאת כך: head == tail
- **isFull** – מתודה בוליאנית שתחזיר אמת אם המערך מלא ואין אפשרות להוסיף עוד ערכים. נבדוק זאת כך: (tail +1) % (capacity+1) == head
- **clear** – מתודה ש"תרוקן" את המערך המעגלי מערכים. אין צורך לנקות את ערכי המערך אלא מספיק לאפס את 2 האינדקסים של ראש וזנב המערך.
- **addNext** – מתודה להכנסת ערך למערך המעגלי המטודה תקבל כפרמטר מספר ותכניס אותו לזנב המערך המעגלי
  לפני כל הכנסה יש לבדוק שהמערך לא מלא
  מכניסים את הערך למקום הפנוי הנוכחי (tail) ומקדמים את tail להצביע על המקום הפנוי הבא להכנסה.
  על מנת שהקידום יהיה מעגלי לא מספיק לקדם את tail באחד, אלא יש לקדם את tail בצורה מעגלית כך: tail = (tail +1) % (capacity+1)
  במידה והמערך לא מלא. במידה והמערך מלא יודפס Vector is full ולא ישנה את המערך בכלל.
- **removeFirst** – מתודה להוצאת ערך מראש המערך המעגלי
  לפני כל הוצאה יש לבדוק שהמערך לא ריק
  יש להוציא את הערך מהמקום עליו מצביע head ומקדמים את head להצביע על הערך הבא להוצאה. (ולא מנקים את הערך שהוצא מהמערך, אלא הוא נשאר כערך "זבל").
  על מנת שהקידום יהיה מעגלי לא מספיק לקדם את head באחד, אלא יש לקדם את head בצורה מעגלית כך:
  head = (head +1) % (capacity+1)
  המתודה תחזיר את הערך שאותו היא הוציאה מהמערך. במידה והמערך ריק יודפס Vector is empty ולהחזיר את הערך 1- (מינוס 1)
- **firstValue** – מתודה להחזרת הערך שנמצא בראש המערך המעגלי, מבלי להוציא אותו. במידה והמערך ריק יודפס Vector is empty ויוחזר הערך 1- (מינוס 1)

- **print** – מתודה שתדפיס למסך את רשימת הערכים במערך המעגלי, מהראש עד לזנב. הערכים יופרדו עם רווח ביניהם ולאחר ההדפסה כולה יש לרדת שורה.

ד. בקובץ התוכנית הראשית, הגדר **פונקציה גלובאלית** בשם **input** שמטרתה לייצר **ולהחזיר RoundVector חדש**, לאחר קליטת ערכים לתוכו.

- הפונקציה תקבל כפרמטר את המספר המקסימלי של ערכים (capacity).
- תגדיר מערך מעגלי חדש על פי הפרמטר שהתקבל.
- תדפיס למסך בקשה להזנה של capacity מספרים ברצף. לדוגמא אם capacity=8 אז הפונקציה תדפיס: Enter 8 numbers:
- תבצע קליטה של capacity ערכים מהמסך ותכניס אותם לפי הסדר למערך המעגלי.
- ותחזיר את המערך החדש

נתון במודל קובץ בשם Vector1Main.cpp המכיל תוכנית הראשית (main) שמטרתה לבדוק נכונות המחלקה שכתבת, הורד את הקובץ ובדוק את תקינות התוכנית שלך (יש להגיש את התוכנית הראשית ביחד עם הקוד שכתבת).

לצורך המחשה, צירפנו בעמוד הבא דוגמא של הפעלת פעולות על מערך מעגלי.
יש לקרוא את הדוגמא משמאל לימין, שורה אחר שורה.

לאחר מכן תוכלו לראות את דרישות שאלה 2 עצמה.

מערך מעגלי של עד 8 ערכים מסוג char
מספר האיברים המקסימלי בתור capacity = 8
גודל פיזי של המערך capacity + 1 = 9