סדנא ב- C++ – 150018

תרגיל בית מספר 2

מחלקות עם שטחים דינאמיים

<u>שים/י לב:</u>

א. הקפד/י על קְרִיאוּת התכנית ועל עימוד (Indentation).
ב. הקפד/י לבצע בדיוק את הנדרש בכל שאלה.
ג. בכל אחת מהשאלות יש להגדיר פונקציות במידת הצורך עבור קריאות התכנית.
ד. יש להגיש את התרגיל על פי ההנחיות להגשת תרגילים (המופיע באתר הקורס) וביניהם: השתמש/י 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 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. הגדר/י מחלקה Point לייצוג נקודה במישור.

המחלקה תכלול את השדות הבאים:

- x – מספר שלם המייצג את המיקום על ציר x
- y – מספר שלם המייצג את המיקום על ציר y

הוסף/י למחלקה לפחות את הבנאים הבאים:

- **empty constructor** - מאתחל את השדות x ו- y באפס (כך שנגדיר נקודה בראשית הצירים).
- **constructor** – המקבל שני פרמטרים ומאתחל את שדות x ו-y לפי הפרמטרים שקבל.
- **copy constructor**
- *(שאלה למחשבה – למה אין פה פונקציה הורסת?)*

הוסיף/י למחלקה את המתודות הבאות:

- הצבה ואחזור (get/set) לכל שדה.
- מתודה המחשבת ומחזירה את המרחק בין שתי נקודות.

ב. הגדר/י מחלקה Polygon לייצוג מצולע כלשהו במישור.

המחלקה תכלול את השדות הבאים:

- מצביע למערך של קודקודים (נקודות במישור – לפי מה שהגדרת בסעיף א).
- מספר הקודקודים במצולע.

הוסף/י למחלקה לפחות את הבנאים ופונקציה הורסת הבאים:

- **empty constructor** - המאתחל את השדה של מספר הנקודות ל- 0 ואת המצביע למערך ל nullptr.
- **constructor** – המקבל כפרמטר את מספר הקודקודים במצולע, ובונה עצם עם מערך בגודל מתאים
- **copy constructor**
- **destructor**

הוסף/י למחלקה לפחות את המתודות הבאות:

- פונקציה אחזור (get) לכל שדה. עבור get של המערך חובה לבנות מערך חדש ולהחזיר את המערך החדש.
- *(שאלה למחשבה – למה אין פה פונקציות set?)*
- מתודה addPoint המקבלת פרמטר אחד מסוג Point ופרמטר שני שהוא אינדקס. על המתודה לשמור הקודקוד זה במערך הקודקודים באינדקס הנשלח כפרמטר.
- 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 הפונקציה להחזיר מספר עשרוני.
- מתודה בוליאנית המקבלת מצולע ובודקת האם המצולע שהתקבל והמצולע הנוכחי זהים. 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)

ג. לצורך הבנה עמוקה של מתודות ה-ctor עליך להוסיף את ההדפסות הבאות:

- ב- empty constructor יש להדפיס : in empty constructor
- ב- constructor שקיבל פרמטר אחד יש להדפיס :
  in one parameter constructor
- ב- copy constructor יש להדפיס: in copy constructor
- ב-destructor יש להדפיס : in destructor

ד. כתוב/י תכנית ראשית הקולטת נתונים על שני מצולעים ומדפיסה את היקפם, **מעוגל למספר השלם הקרוב ביותר**, (אפשר להשתמש בפונקציה round) באופן הבא: במידה והמצולעים זהים יש להדפיס equal ואת היקפם. במידה ואינם זהים יש להדפיס not equal ועבור כל אחד מהם את היקפו.

התכנית תדפיס עבור כל מצולע: enter number of sides: ותקלוט את מספר הצלעות.
לאחר מכן התכנית תדפיס: enter the point values: ותקלוט את שיעורי הנקודות.
הקלט יהיה מהצורה: (x1,y1) (x2,y2) … (xN,yN)
כאשר: N הוא מספר הקודקודים במצולע, ו: xi, yi הם הקואורדינאטות של הנקודות במצולע.
בכל מקרה של קלט לא תקין התכנית תדפיס ERROR.

דוגמאות להרצת התכנית:

ריבוע שצלעו 2 ומשולש:

```
enter number of sides:
4
in one parameter constructor
enter the point values:
(0,0) (0,2) (2,2) (2,0)
enter number of sides:
3
in one parameter constructor
enter the point values:
(1,1) (2,0) (3,1)
in copy constructor
in destructor
not equal
perimeter: 8
perimeter: 5
in destructor
in destructor
```

משולשים ישרי זווית שצלעותיהם 3-4-5:

```
enter number of sides:
3
in one parameter constructor
enter the point values:
(10,10) (10,14) (13,10)
enter number of sides:
3
in one parameter constructor
enter the point values:
(13,10) (10,10) (10,14)
in copy constructor
in destructor
equal
perimeter: 12
in destructor
in destructor
```

תזכורת: בכדי להשתמש בפונקציות מתמטיות יש להכליל את הספריה cmath.

שאלה מס' 2:

א. הגדר/י מחלקה בשם Vector למימוש מערך של מספרים שלמים באורך כלשהו.
שימו לב – המחלקה שונה מעט מהמחלקה שהוצגה בכיתה.
גודל הוקטור יהיה חזקה של 2.
יש לשים לב לשינוי הזה במיוחד:

- בבנאי פרמטר של המחלקה (המקבל גודל מערך),
- במתודה לשרשור שני מערכים (strnewcat)
- ובמתודה של הכנסת ערך למערך (insert)

בבנאי ובשרשור – יש לבנות וקטור חדש. אם גודל המערך הוא כבר חזקה של 2 אז אין בעיה, אבל אם לא, אז יש למצוא את החזקה של 2 הקרובה מלמעלה (גדולה) מהגודל הנדרש.
ניתן להשתמש בנוסחה הבאה:

```
capacity = pow(2, ceil(log(requested_size) / log(2)))
```

כך שה requested_size הוא גודל הנדרש

במתודה של insert תצליחו לשמור על גודל של חזקה של 2 על ידי הכפלה גודל המערך ב-2.

המחלקה תכלול את השדות הבאים:

- data - מצביע למערך של שלמים
- capacity - אורך הוקטור (מספר האיברים המקסימלי האפשרי כרגע במערך).
- size - מספר הערכים הנמצאים בפועל בוקטור

ב. הוסף/י למחלקה לפחות את הבנאים ופונקציה הורסת הבאים:

- **constructor -** מקבל כפרמטר את גודל הוקטור המבוקש ויוצר אותו. אם לא התקבל ערך, הקונסטרקטור יציב כערך ברירת מחדל 2.
  אם כן התקבל מספר כפרמטר, על הבנאי לוודא שהמספר שהתקבל הוא חזקה של 2. אם המספר שהתקבל כפרמטר אינו חזקה של 2, על המתודה למצוא את המספר שהוא כן חזקה של 2 שהוא יותר גדול והכי קרוב למספר שהתקבל כפרמטר. (יש לעין בנוסחה לעיל)
  שימו לב: לאחר שמאתחלים השדה capacity, צריכים לבנות את המערך data בגודל המתאים ולאתחל את השדה size ל-0.

- **copy constructor**
- **destructor**

ג. הוסף/י למחלקה לפחות את המתודות הבאות:

- מתודה בשם ()getCapacity המחזירה את מספר האיברים <u>המקסימלי</u> בוקטור בעת הקריאה למתודה .
- מתודה בשם ()getSize המחזירה את מספר האיברים הקיימים <u>בפועל</u> בוקטור.
- *(שאלה למחשבה – למה אין פה פונקציות set?)*

- מתודה בשם ()print. המתודה תדפיס את נתוני הוקטור: גודל מקסימלי, גודל בפועל, והערכים בוקטור.
לדוגמא וקטור באורך 4 שבו שני תאים מלאים כרגע (5,19) יודפס כך:
capacity: 4 size: 2 values: 5 19
- מתודה בשם assign (מתפקדת כאופרטור =) המקבלת כפרמטר וקטור נוסף ומשנה את הוקטור הנוכחי להיות זהה לוקטור שהתקבל.
- מתודה בשם isEqual (מתפקדת כאופרטור הבוליאני ==) המקבלת כפרמטר וקטור נוסף ומחזירה true כאשר שני הוקטורים זהים הן בגודלם הנוכחי המלא (size) והן בתכנם. (לא משווה בין capacity). אחרת תחזיר false. (בכל אינדקס במערך – ישנו ערך זהה לערך הנמצא באינדקס זה במערך השני). כלומר עבור שני המערכים הבאים:
[3, 5, 7, 9] , [5, 3, 7, 9]
המתודה תחזירה false
- מתודה בשם at המקבלת אינדקס (מתפקדת כאופרטור []) **להצבה והחזרה** של ערך בתא מבוקש בוקטור. אם התא המבוקש אינו בטווח התאים המלאים במערך יש להדפיס ERROR ולהחזיר כפרמטר את האיבר הראשון. כותרת המתודה: int& at(int index);
- מתודה בשם scalmul המקבלת כפרמטר וקטור נוסף ומבצעת מכפלה סקלרית בין שני וקטורים (יש להכפיל ערך כל תא בוקטור הראשון עם הערך בתא המקביל בווקטור השני ולסכום את התוצאות). אם בשני הוקטורים אין מספר זהה של ערכים יש להדפיס ERROR ולהחזיר 1-.
- מתודה בשם strnewcat לשרשור שני וקטורים.
המתודה מקבלת וקטור כפרמטר. על המתודה לשרשר בין שני הוקטורים כך שבשדה data של הוקטור החדש יופיעו: ראשית - הערכים השמורים בשדה data של הוקטור שבעצם שזימן את המתודה ולאחריהם - הנתונים השמורים בשדה data של הוקטור שנשלח כפרמטר. size של הוקטור החדש יהיה כמובן מספר האיברים שהיו בשני הוקטורים.
רוצים מצד אחד שה-capacity של וקטור החדש יהיה סכום של ה-capacity של הוקטור שקרא לפונקציה ושל הוקטור שהתקבל כפרמטר. מצד שני, חובה שה-capacity הינו מספר שהוא חזקה של 2. יש לעניין בנוסחה לעיל כדי לחשב את ה capacity החדש.
- מתודה בשם ()clear שתפקידה לרוקן את וקטור (ריקון **ערכים**).
- מתודה בשם ()delLast המוחקת את האיבר האחרון בוקטור. במידה והוקטור ריק(חמיקה) יש להדפיס ERROR.
- מתודה בשם insert(int val) המציבה את הערך val במקום הבא הפנוי במערך. במידה והמערך מלא יש להגדיל את המערך(באופן דינאמי) פי שניים מגודלו הנוכחי, יש להעתיק את הערכים מהמערך המקורי למערך המוקצה החדש ולשחרר את המערך המקורי מהזיכרון. (כלומר - כאשר size מגיע לגודל ה-capacity יש להגדיל את המערך פי שניים מגודלו בזמן ההוספה - ולבצע העתקה ושחרור בהתאם).

יש להשתמש בתכנית הראשית הבאה בכדי לבחון את נכונות המתודות שכתבת:

```
#include "Vector.h"
#include <iostream>
using namespace std;
enum options
```

```
{
	STOP, ASSIGN, IS_EQUAL, SCALAR_MULTIPLY , ADD, CLEAR, DELETE_LAST,
AT, INSERT
};
int main()
{
	cout << "Test 1 - Constructors" << endl << endl;
	Vector  v1(4), v2(10), v3;
	cout << "v1 capacity: " << v1.getCapacity()
		<< " v1 size: " << v1.getSize()<< endl;
	cout << "v2 capacity: " << v2.getCapacity()
		<< " v2 size: " << v2.getSize() << endl;
	cout << "v3 capacity: " << v3.getCapacity()
		<< " v3 size: " << v3.getSize() << endl;

	cout << endl << "Test 2 - Assign" << endl << endl;
	for (int i = 1; i <= 4; i++)
	{
		v1.insert(i);
		v2.insert(i);
		v3.insert(i + 4);
	}

	cout << "v1 capacity: " << v1.getCapacity()
		<< " v1 size: " << v1.getSize()<< endl;
	cout << "v2 capacity: " << v2.getCapacity()
		<< " v2 size: " << v2.getSize() << endl;
	cout << "v3 capacity: " << v3.getCapacity()
		<< " v3 size: " << v3.getSize() << endl;

	int choice, val, index;
	cout << endl << "Test 3 - Operations" << endl << endl;
	cout << "enter your choice 0-8:\n";
	cin >> choice;
	while (choice)
	{
		switch (choice)
		{
		case ASSIGN: v3.assign(v1);
			break;
		case IS_EQUAL:
			if (v1.isEqual(v2))
				cout << "v1==v2\n";
			else
				cout << "v1!=v2\n";

			if (v1.isEqual(v3))
				cout << "v1==v3\n";
			else
				cout << "v1!=v3\n";
			break;
		case SCALAR_MULTIPLY:
			cout << "v1*v2=" << v1.scalmul(v2) << endl;
			break;
		case ADD:
			v3.assign(v1.strnewcat(v2));
			break;
		case CLEAR:
			v1.clear();
			break;
		case DELETE_LAST:
			v2.delLast();
			break;
```

```
                case AT:
                        cout << "enter index:" << endl;
                        cin >> index;
                        cout << "enter value:" << endl;
                        cin >> val;
                        v3.at(index) = val;
                        break;
                case INSERT:
                        cout << "enter value:" << endl;
                        cin >> val;
                        v3.insert(val);
                        break;
                default: cout << "ERROR";
                }
                v1.print();
                v2.print();
                v3.print();
                cout << endl << "enter your choice 0-8:\n";
                cin >> choice;
        }
        return 0;
}
```

דוגמאות להרצת התכנית:

```
Test 1 - Constructors

v1 capacity: 4 v1 size: 0
v2 capacity: 16 v2 size: 0
v3 capacity: 2 v3 size: 0

Test 2 - Assign

v1 capacity: 4 v1 size: 4
v2 capacity: 16 v2 size: 4
v3 capacity: 4 v3 size: 4

Test 3 - Operations

enter your choice 0-8:
3
v1*v2=30
capacity: 4 size: 4 values: 1 2 3 4
capacity: 16 size: 4 values: 1 2 3 4
capacity: 4 size: 4 values: 5 6 7 8

enter your choice 0-8:
0
```

```
Test 1 - Constructors

v1 capacity: 4 v1 size: 0
v2 capacity: 16 v2 size: 0
v3 capacity: 2 v3 size: 0

Test 2 - Assign

v1 capacity: 4 v1 size: 4
v2 capacity: 16 v2 size: 4
v3 capacity: 4 v3 size: 4

Test 3 - Operations

enter your choice 0-8:
4
capacity: 4 size: 4 values: 1 2 3 4
capacity: 16 size: 4 values: 1 2 3 4
capacity: 32 size: 8 values: 1 2 3 4 1 2 3 4

enter your choice 0-8:
0
```