סדנא ב- C++ – 150018

תרגיל בית מספר 5

מחלקה מוכלת – רשימה לינארית

שים/י לב:

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.
השתמש/י בקבועים מתי שאפשר.

ה. יש לתעד את התכנית גם עבור פונקציות אותם הנך מגדיר/ה וכן על תנאים ולולאות וקטעי קוד מורכבים,
ו. **יש** להעביר את האובייקטים הפרמטרים ב-cbr (הפניה) כמה שיותר. (הרבה יותר יעיל לשלוח אובייקט כ cbr כי ככה הוא לא קורא לבנאי העתקה וזה חוסך גם זמן ריצה וגם זכרון)
לדוגמה בשאלה 1 כשכותבים את החתימה למתודה =operator עדיף אם יראה ככה:

```
List & List::operator=(const List& l)
```

ז. יש להקפיד על טיפוס מוחזר ממתודה. אם המתודה לא אמורה לחשב משהו אז עדיף לכתוב void בחתימה. אם המתודה אמורה להחזיר אובייקט יש לוודא האם היא מחזירה אובייקט או האם היא מחזירה הפנייה לאובייקט ולא להתבלבל ביניהם (פרטים בשאלה 2).
ח. ובנוסף, **דוגמת הרצה לכל תכנית בסוף הקובץ!**
הגשה בזוגות – שימו לב: אחד מגיש את התרגיל עם שמות של שני הסטודנטים. אם אתם נמצאים בשתי קובצות שונות אז גם להוסיף מספר קבוצה לשני הסטודנטים. השני מגיש קובץ וורד עם שמות של שני הסטודנטים.

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

שאלה מס' 1:

השלם/י את הגדרת המחלקה List שהוצגה בהרצאה, כך שתאפשר לטפל ברשימה המכילה איברים ממוינים בסדר לא עולה (כל איבר ברשימה קטן או שווה מהאיבר הקודם לו).
הוסף/י את המתודות הבאות:

א. =operator - העתקה עמוקה של רשימה אחת לתוך השניה בסדר זהה.
ב. מתודת פלט - >> operator להדפסת כל אברי הרשימה כל האיברים באותה שורה עם רווח אחד שמפריד ביניהם.
ג. מתודת קלט - << operator לקליטת אברי המערך בצורה ממוינת בסדר יורד, כלומר, הקלט יסתיים כאשר יתקבל ערך שאינו קטן (ממש) מהערך הקודם לו. (אין איברים כפולים בשלב הזה.)

ד. insert (int key) מתודה המקבלת מספר שלם key ומכניסה אותו למקום המתאים ברשימה הממוינת. השלב הזה שומר על הסדר **לא עולה** (ייתכנו איברים כפולים).

ה. מתודת מחיקה – remove (int key). המתודה מקבלת מספר שלם key. במידה והערך key קיים ברשימה, המתודה מוחקת אותו מהרשימה תוך שמירה על המיון של הרשימה. במידה והערך לא קיים, תשלח הודעת חריגה "value not found". במידה והערך קיים יותר מפעם אחת, יש למחוק את המופע הראשון שנתקלים בו.

נתונה התכנית הראשית הבאה, הבוחנת את נכונות המחלקה חובה עליכם להוסיף try ו- catch במקום המתאים:

```
#include <iostream>
using namespace std;
#include "List.h"

enum CHOICES { EXIT, INSERT, REMOVE, ASSIGN, PRINT };

int main(){
	List lst;
	List lst2;
	int choice, val;

	cout << "enter the list values\n";
	cin >> lst;
	cout << "choose 0-4\n";
	cin >> choice;
	while (choice != EXIT){
		switch (choice) {
			case INSERT:
				cout << "enter a value to insert\n";
				cin >> val;
				lst.insert(val);
				break;
			case REMOVE:
				cout << "enter a value to remove\n";
				cin >> val;
				lst.remove(val);
				break;
			case ASSIGN:
				lst2 = lst;
				cout << "list 2: " << lst2;
				cout << "list 1: ";
			case PRINT:
				cout << lst << endl;
				break;
			default:cout << "ERROR\n";
		}
		cout << "choose 0-4\n";
		cin >> choice;
	}
	return 0;
}
```

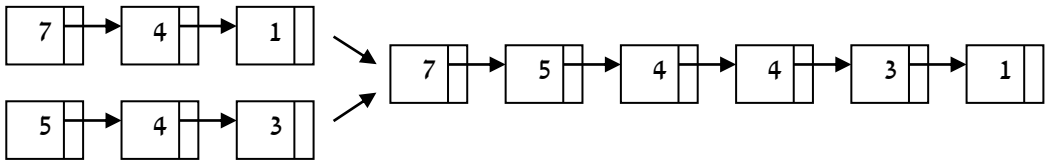
דוגמה להרצת התכנית הרצה:

```
enter the list values
25 20 15 8 6 10
choose 0-4
4
25 20 15 8 6
choose 0-4
1
enter a value to insert
15
choose 0-4
1
enter a value to insert
7
choose 0-4
4
25 20  15 15 8 7 6
choose 0-4
2
enter a value to remove
15
choose 0-4
2
enter a value to remove
8
choose 0-4
3
list2: 25 20 15 7 6
list1: 25 20 15 7 6
choose 0-4
0
```

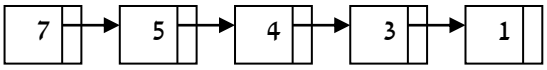
שאלה מס' 2:

השתמש/י במחלקה List המטפלת ברשימה ממוינת (המוגדרת לעיל), וכתוב/י את שלושת הפונקציות ה**גלובליות** הבאות (שים/י לב – אין ליצור פונקציות חברות של המחלקה, אלא ליצור את הפונקציות ב-main ולהיעזר במתודות public בלבד):

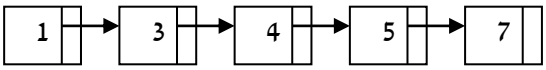
א. פונקציה merge למיזוג שתי רשימות. הפונקציה מקבלת שתי רשימות ממוינות lst1 ו – lst2 מטיפוס List, ומחזירה רשימה חדשה הבנויה ממיזוג שתי הרשימות בסדר **לא עולה**. (שים/י לב – ברשימה החדשה ייתכנו איברים כפולים)



ב. פונקציה makeSet להפיכת רשימה לקבוצה. הפונקציה מקבלת רשימה ממוינת בסדר **לא עולה**. הפונקציה משנה את הרשימה, כך שכל איבר יופיע בה פעם אחת בלבד (לאחר ביצוע הפונקציה, לא יהיו ברשימה איברים כפולים – הרשימה תהיה ממויינת בסדר **יורד**).



ג. פונקציה reverse ההופכת את הרשימה, כלומר האיבר האחרון יצביע לאיבר שלפניו וכו' עד כאשר האיבר הראשון יצביע ל-null. (מה שיגרום לרשימה ממויינת בסדר לא יורד)



נתונה התכנית הראשית הבאה, הבוחנת את נכונות הפונקציות:

```
#include <iostream>
#include "List.h"
using namespace std;
//הגדרת ומימוש הפונקציות

int main()
{
	List lst1, lst2, mergedList;

	cout<<"enter sorted values for the first list:"<< endl;
	cin>>lst1;
	cout<<"enter sorted values for the second list:"<< endl;
	cin>>lst2;

	mergedList = merge(lst1,lst2);
	cout <<"the new merged list: " << mergedList <<endl;

	makeSet(mergedList);
	cout<<"the new merged set: " << mergedList << endl;
```

```
        reverse(mergedList);
        cout<<"the new merged reverse: " << mergedList << endl;


        return 0;
}
```

הערות כלליות לגבי שאלה 2:

1. אפשר להשתמש במחלקה של List שנכתבה בשאלה 1.
2. לפי הוראות השאלה – כותבים רק את הקוד של שלושת הפונקציות. **אסור לשנות שום דבר** במחלקה List כדי לענות על שאלה זו
3. אין גישה בכלל לשדות הפרטיים של ה List (כלומר אין דרך לגעת ב head וכל מיני דברים אחרים)
4. הקוד ייכתב תוך כדי שימוש במתודות של המחלקה List בלבד.

רמז עבור הפונקציות שלא מחזירות ערך (בתרגיל שלנו פונקציות makeset ו-reverse)

- הפונקציה מקבלת פרמטר מסוג אובייקט (רשימה) ועל הפונקציה לשנות את האובייקט
- דרך אחת לעשות את זה היא :
  - לבנות רשימה חדשה (אובייקט חדש),
  - להפעיל מתודות של המחלקה List על האובייקט החדש ו/או האובייקט שהתקבל כפרמטר
  - להציב את האובייקט החדש בתוך האובייקט שהתקבל כפרמטר

הבהרה עבור הפונקציה המחזירה אובייקט (בתרגיל שלנו הפונקציה merge)

- תזכורת :
  - כשמחזירים אובייקט, אזי באופן אוטומטי נקרא בנאי העתקה של המחלקה לפני שזה הורס את האובייקט
  - כשמחזירים הפנייה לאובייקט זה לא קורא לבנאי העתקה לפני שזה הורס את האובייקט. ולכן מחזיר כתובת עם ערך זבל.
- בתרגיל הזה:
  - אם הפונקציה merge תחזיר הפנייה לאובייקט מסוג List אז כשהיא תחזור לתוכנית הראשית תקבלו שגיאה (הפונקציה תחשב נכון את הרשימה החדשה שמיזוג בין 2 רשימות ואז בסוף הפונקציה תחזיר כתובת של הרשימה החדשה, תהרוס את הרשימה החדשה ולכן התוכנית הראשית תקבל כתובת של זבל.)
  - אם הפונקציה merge תחזיר אובייקט מסוג List אז הכל יהיה בסדר (הפונקציה תחשב נכון את הרשימה החדשה שמיזוג בין 2 רשימות ואז בסוף תקרא לבנאי העתקה, תהרוס את הרשימה החדשה ותחזיר את העותק שבנאי העתקה בנה. התוכנית הראשית תקבל העתק.)
  - לכן מומלץ שחתימת הפונקציה של merge היא:

```
List merge(const List &lst1, const  List &lst2);
```

דוגמה להרצת התכנית:

```
enter sorted values for the first list :
6 5 4 3 2 1 9
enter sorted values for the second list :
7 5 4 3 9
the new merged list : 7 6 5 5 4 4 3 3 2 1
the new merged set : 7 6 5 4 3 2 1
the new merged reverse : 1 2 3 4 5 6 7
```


בהצלחה!