המסלול האקדמי
המכללה למינהל

להצלחה יש דרך

מבחן לדוגמא

ד"ר אליהו חלסצ'י
eliahukh@colman.ac.il

הקדמה

במבחן זה עליכם לענות על 3/3 שאלות תכנותיות ב ++C. משך המבחן 3 שעות. חומר פתוח. עליכם להקפיד היטב על ההוראות, ובפרט על הוראות ההגשה, שכן הבדיקה הינה אוטומטית.

אתם מקבלים:

- שלושה קובצי h, אחד לכל שאלה.
- אין להוסיף include-ים כלשהם וגם אין בכך צורך.
- עליכם להשלים את הקוד בקובצי ה h בלבד, ע"פ הגדרות השאלה
- בכל סוף קובץ h ישנה פונקציית mainAPI לבדיקת ה API של המחלקות השונות.
  - היא בודקת את כל ה API הנדרש לשאלה ☺
  - באמצעותה תוכלו לדעת שאין לכם טעויות API שיגרמו לשגיאות קומפילציה
  - תוכלו כמובן להוסיף לה בדיקות נוספות משלכם
- בנוסף אתם מקבלים main לוקאלי שפשוט קורא לשלוש פונקציות הבדיקה של ה API.
- לעומתו, mainTrain מכיל בדיקות לוגיות. זהו ה main של מוד האימון. הוא חשוף, וייתן לכם מושג לגבי הבדיקות הלוגיות של המבחן.
  - שימו לב שהוא אינו בודק את *כל* הבדיקות הלוגיות של המבחן, לכן עליכם להתרכז בהוראות המבחן ולא רק לגרום ל mainTrain להצליח.

עליכם להגיש:

- את קובצי ה h מושלמים. לא ב zip או דומיו, אלא את קובצי המקור עצמם.

קוד שלא מתקמפל או שיש לו שגיאות בזמן ריצה יקבל הפחתה אוטומטית של 10 נקודות.

משוב למוד האימון הוא מלא ותוכלו להגיש אליו בעת המבחן כמה פעמים שרק תרצו.

הגשת המבחן נעשית למוד הגשה בלבד. מי שלא הגיש לשם דינו כמי שלא הגיש את המבחן. יש להגיש לשם את קובצי המקור עד לסוף זמן המבחן. כל עוד יש לכם שגיאות קומפילציה או ריצה תוכלו לתקן ולהגיש שוב. אך ברגע שאין יותר שגיאות כאלו \ תם זמן המבחן ניתן להגיש רק פעם אחת. למוד הגשה לא יינתן פידבק. הציונים למבחן יינתנו לאחר ביקורת שלי.

עליכם להגיע לפני תחילת המבחן ולהתיישב בעמדות שבמעבדה. במידה והנכם רשומים למעבדה 36/7 אנא שבו בצד של המעבדה אליה אתם רשומים. עם תחילת המבחן יתפרסם טופס המבחן במודול של הקורס. האינטרנט ינותק והמבחן יתחיל. אל תחכו לרגע האחרון להגיש את המבחן בלחץ, ואז לגלות ששכחתם משהו. תכננו את הזמן היטב.

המבחן מתרכז באלמנטים הבאים:

- הגדרת טיפוסים נכונה – ירושה, הכלה, ירושה מרובה, בנייה והריסה.
- שימוש נכון בפולימורפיזם לאלג' גנרי ומבנה נתונים גנרי
- תכנות גנרי באמצעות templates, object functions
- אופרטורים
- קבצים
- STL

הערה: המבחן רק נראה ארוך בגלל שהדבקתי את קובצי קוד המקור המצורפים, לא להיבהל ☺

בהצלחה!

אלי

שאלה 1: (33 נק')

ברצוננו לממש רשימה חד-כיוונית, מעגלית. הקוד הבא, הנמצא בקובץ RoundList.h מגדיר את הרשימה המעגלית באופן הבא:

- המחלקה RoundList<T>, ובתוכה
  - המחלקה הפרטית Item<E>
  - המחלקה הפומבית Iterator

ברשימה מעגלית לכל איבר יש מצביע לאיבר הבא, כאשר האיבר האחרון מצביע לאיבר הראשון. עליכם להשלים את החסר בקוד על מנת לעמוד בדרישות הבאות:

- הבנאי של RoundList יקבל אובייקט מסוג T כלשהו, ובאמצעותו יאתחל את המשתנה head, ואת גודל הרשימה להיות 1. כמו כן, האיבר הבא של head יהיה head עצמו.
- לאיטרטור השלימו את האופרטורים ++ ו *
- המתודה begin של RoundList תחזיר איטרטור המצביע לאיבר הראשון ברשימה
- המתודה insert של RoundList תקבל אובייקט מסוג T ואיטרטור, ותכניס (העתק של) האובייקט **מימין** לאיטרטור.

```
#include <iostream>
using namespace std;

template<class T>
class RoundList{

	template<class E>
	class Item{
		_________ next;
		E data;
	public:

		Item() :next(NULL){}
		Item(const E& pdata) :next(NULL), data(______){}

		void setNext(________ next){ this->next = next; }
		_________ getNext(){ return next; }
		void setData(__________){ this->data = data; }
		________ getData(){ return data; }
	};

	Item<T>* head;
	int size;

public:

	RoundList(______) :head(____), size(__){ ___________________ }

	class Iterator{
		________ p;
	public:
		Iterator(_______ pt = NULL) :p(__){}
		Iterator& operator++(int){

			__________
			return ________;
		}
		T& operator*(){ _________________ }
		friend class _________________;
	};
```

```
        Iterator begin() { _____________ }
        void insert(const T& newItem, Iterator it){
                _____________________________
                _____________________________
                _____________________________
                _____________________________
                size++;
        }

        int getSize(){return size;}

};

// API test
void mainAPIRoundList()
{
        RoundList<int> list(5);
        for (int i = 1; i<5; i++)
                list.insert(i, list.begin());

        RoundList<int>::Iterator it = list.begin();
        for (int i = 0; i<list.getSize() * 2; i++){
                cout << *it << "->";
                it++;
        }
        cout << endl;
        // output: 5->4->3->2->1->5->4->3->2->1->
}
```

כפי שניתן לראות, המתודה mainAPIRoundList יוצרת רשימה של int חדשה עם הערך 5. לאחר מכן מכניסה את הערכים 1 עד 4, כולם מימין לתחילת הרשימה. לאחר מכן אנו מדפיסים את הרשימה באמצעות האיטרטור. נשים לב שהמשכנו עד ל size*2, ואין זה משנה מפני שהרשימה מעגלית... הפלט הצפוי מופיע כהערה.

בהצלחה.

שאלה 2: (34 נק')

ברצוננו לשמור מערך פולימורפי של סטודנטים בקובץ, ומאוחר יותר לטעון אותו. נרצה להגדיר:

- סטודנט לתואר ראשון
  - לסטודנט יש שם (char*), ממוצע ציונים (double) וציון לפרויקט שנתי (double)
- סטודנט לתואר שני
  - לסטודנט יש שם (char*), ממוצע ציונים (double) ומס' עמודים לתיזה (int)
- סטודנט לתואר שלישי
  - לסטודנט יש שם (char*), ממוצע ציונים (double) וציון סקר הוראה (char)

השלימו את הקוד הבא, הנמצא ב students.h, כדי שהשמירה והטעינה המתבצעת ב mainAPIStudent תעבוד כראוי.

- עליכם להסיק מתוך הקוד הכתוב בפונקציה mainAPIStudent כיצד להשלים את הקוד החסר
- המלצה: תתחילו מהפונקציה mainAPIStudent

```
#include <iostream>
#include <fstream>
#include <string.h>
using namespace std;

class Student{
        ______________
        ______________

protected:
        ______________

public:

        Student() :name(NULL), average(0){}

        Student(const char* name, double average){ /* להשלים */}

        _________ getName(){ return name; }
        _________ getAverage(){ return average; }

        _______ void save(ofstream& out){ /* להשלים */}

        _______ void load(ifstream& in){ /* להשלים */}

        ________ ~Student(){ delete[] name; };

        void saveType(ofstream& out){
                out.write(&type, 1);
        }

        _______ char loadType(ifstream& in){
                char type;
                in.read(&type, sizeof(char));
                return type;
        }
};
```

```
class UndergradStudent ________________{
        ____________________;
public:
        UndergradStudent(){}
        UndergradStudent(const char* name, double average, double projectGrade)
        :_________{ ________________; type = 'U'; }

        ____________________________________________

        ____________________________________________

        ____________________________________________

        ____________________________________________

        ____________________________________________

};

class MAStudent { /* להשלים */};

class PhdStudent { /* להשלים */};

int mainAPIStudent()
{
        int n = 3;
        Student** arr = new Student*[n];
        arr[0] = new UndergradStudent("a", 89.0, 95.5);
        arr[1] = new MAStudent("b", 95.0, 60);
        arr[2] = new PhdStudent("c", 90.0, 'A');

        ofstream out("students.db", ios::binary);
        out.write((char*)&n, sizeof(n));
        for (int i = 0; i<n; i++){
                arr[i]->saveType(out);
                arr[i]->save(out);
                delete arr[i];
        }
        delete[] arr;
        out.close();

        ifstream in("students.db", ios::binary);
        in.read((char*)&n, sizeof(n));
        arr = new Student*[n];
        for (int i = 0; i<n; i++){
                char type = Student::loadType(in);
                switch (type){
                case 'U':arr[i] = new UndergradStudent(); break;
                case 'M':arr[i] = new MAStudent(); break;
                case 'P':arr[i] = new PhdStudent(); break;
                }
                arr[i]->load(in);
                cout << arr[i]->getName() << " " << arr[i]->getAverage() << endl;
        }
        in.close();

        for (int i = 0; i<n; i++){delete arr[i];}
        delete[] arr;

}
```

הפלט יהיה שמות הסטודנטים a,b,c והציונים המתאמים שלהם. בהצלחה.

שאלה 3: (33 נק')

הוסיפו לקוד הנמצא בקובץ stlTest.h, את ה object functions המתאימים כדי שפונקציית ה mainAPISTL תבצע:

- ספירה כמה עובדים נמצאים מתחת לגיל 30
- תמיין את העובדים לפי גילאים מהקטן לגדול
- תדפיס את הרשימה הממוינת

כמו כן, עליכם לממש את הפונקציה הגנרית apply_if אשר מפעילה פונקציה שניתנת כפרמטר, על כל האיברים במבנה נתונים כלשהו במידה והם עונים על תנאי המתקבל כפרמטר.

הפעולה האחרונה תדפיס רק את מי שגילו קטן מ 20.

```
#include <iostream>
#include <string.h>
#include <vector>
#include <algorithm>

using namespace std;

class Employee{
	char* name;
	int age;

public:

	Employee(const char* name, int age){
		this->name = new char[strlen(name) + 1];
		strcpy(this->name, name);
		this->age = age;
	}

	const char* getName()const { return name; }
	int getAge()const { return age; }

};
/* הוסיפו כאן את הקוד החסר הנדרש */
int mainAPISTL()
{
	vector<Employee> emps;
	emps.push_back(Employee("A", 19));
	emps.push_back(Employee("B", 24));
	emps.push_back(Employee("C", 42));
	emps.push_back(Employee("D", 35));

	cout << count_if(emps.begin(), emps.end(), AgeChecker(30)) << endl; // 2

	sort(emps.begin(), emps.end(), AgeComparator());

	// A B D C
	for_each(emps.begin(), emps.end(), PrintName());
	AgeChecker ac(20);
	PrintName pn;
	apply_if(emps.begin(),emps.end(),ac,pn); // A
}
```

בהצלחה

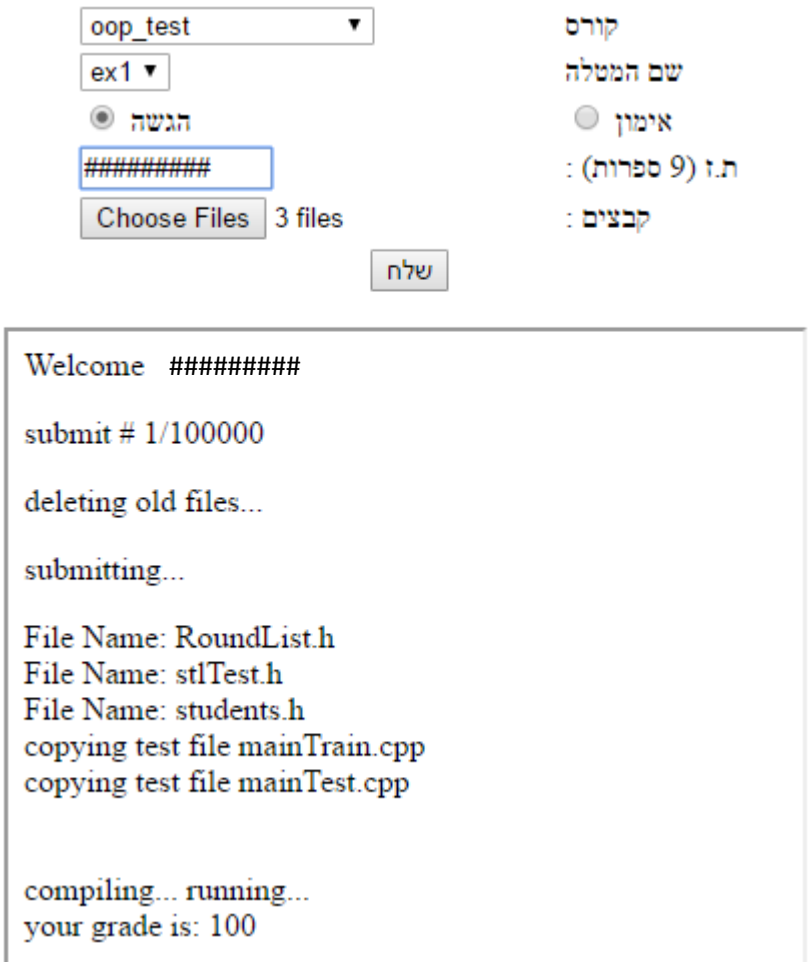
קובץ ה main.cpp לבדיקת ה API מוגדר כך:

```
#include "RoundList.h"
#include "students.h"
#include "stlTest.h"

int main()
{
	mainAPIRoundList();
	mainAPIStudent();
	mainAPISTL();
}
```

מומלץ לעבוד בכל פעם על שאלה אחרת ולהסתיר בהערה את השאר.

הגשה: עליכם להגיש רק את 3 קובצי ה h למערכת הבדיקה, תחת oop_test ותרגיל ex1. מכיוון שזה מבחן לדוגמא תוכלו להגיש אינספור פעמים, אך זכרו, במבחן האמתי תוכלו להגיש במוד הגשה רק כל עוד לא נגמר זמן הבחינה ויש לכם שגיאות קומפילציה או ריצה.

כמו כן, בעת המבחן יהיה לכם לינק לשרת גיבוי להגשת הקבצים, ולינקים לאתרים ברשת להם נרשה לכם להיכנס.

בהצלחה!