Advanced Programming

Homework Assignment 6

Inheritance

General guidelines:

a. Maximize readability and ensure indentation.
b. Do exactly as required in the questions.
c. Every class in every question must be submitted in two seperate files – a header file (.h) and an implementation file (.cpp).
d. Add functions and helper methods as necessary to ensure readability.
e. Submission must be done according to the submission guidelines – a document published in the course website – read them carefully!
f. Use relevant names.
g. Use comments to document your functions and complex code parts. **Add an output example at the end of your file!**
h. Individual work and submission – paired submission is not allowed.

Important note: Unless otherwise specified, ever homework has no more than one week for submission.
Open submission boxes past the deadline are not permission for late submission.

Question 1:

A **Round List** is defined as a linked list in which the next pointer for the last link contains the address of the head. The methods available for the round list are the same methods that are available for a linked list, and also the following methods:

- **addToEnd(int)** – receives a value and adds it as a new element to the end of the list.
- **search(int)** – receives a non-negative index i and returns the value of the i-th element in the list. If i is greater than the number of elements in the list, the iteration over the list repeats in a circular fashion until i elements are iterated (there is no overflow). The first element is of index 0. If the list is empty the method returns -1.

Declare and implement a class **RoundList** that inherits from List. (The class List to be used is to be taken from the Moodle in the General tab – there is a directory – מבנה נתונים קוד). Implement all the necessary methods, taking into account the following instructions:

- Remove the method search from the class List. In this exercise search is used for a different purpose.
- Do not add any fields. The only field allowed is the address of the head link that already exists in the base class List.
- It is up to you to determine what methods from List must have an explicit override in RoundList, in order to make sure all the methods do as required.
- Make sure to implement in RoundList the additional two methods described above.

In order to test your code, use the following main program, **exactly as it is presented** (no changes are allowed):

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

enum CHOICES{
	EXIT, ADD, ADD_TO_END, REMOVE_FIRST, SEARCH, CLEAR, EMPTY
};
int main(){
	RoundList ls1;
	int choice;
	cout << "Enter your choice: ";
	cin >> choice;
	while(choice != EXIT)
	{
		int num;
		switch(choice){
			case ADD :  cout << "Enter 5 numbers: ";
					for(int i=0; i < 5; i++)
					{
						cin >> num;
						ls1.add(num);
					}
					break;
			case ADD_TO_END :cout << "Enter 5 numbers: ";
						for(int i=0; i < 5; i++)
						{
							cin >> num;
							ls1.addToEnd(num);
						}
						break;
			case REMOVE_FIRST : ls1.removeFirst();
						break;
			case SEARCH: cout << "Enter a number: ";
					cin >> num;
					cout << ls1.search(num)<<endl
									break;
			case CLEAR: ls1.clear();
					break;
			case EMPTY: if(ls1.isEmpty())
						cout << "Empty"<<endl;
					else
						cout << "Not empty" << endl;
					break;
			default: cout<< "ERROR!"<<endl;
		}
		cout << "Enter your choice: ";
		cin >> choice;
	}
	return 0;
}
```

Question 2:

The object of this exercise is implementation of a grant managing system for a higher education institution. In the institution study students for bachelor, master and doctorate level degrees.

Design the following hierarchy of classes that represents the students in the institution:

a. **Student** – and abstract base class that represents a student.
The class must contain fields for the following data: student id, first name, surname, the number of courses the student is studying.
The class must contain the following methods:
   - constructor – initializes the fields.
   - print() – prints the student info. For each field print a separate line containing the description of the field followed by its value. The format and exact wordings can be seen in the example below.

b. **BA** – a class that inherits from **Student** and represents a bachelor level student.
The class must contain apart from inherited fields also fields for the following data: An array of grades and its size.
The class must contain a boolean method milga() that returns true if the student is eligible to receive a grant. The grant criteria are described below.

c. **MA** – a class that inherits from **BA** and represents a master level student.
The class must contain apart from inherited fields also fields for the following data: a boolean determining if the student is writing a thesis.
The class must contain a boolean method milga() that returns true if the student is eligible to receive a grant. The grant criteria are described below.

d. **PHD** – a class that inherits from **Student** and represents a doctorate level student.
The class must contain apart from inherited fields also fields for the following data: the number of weekly hours the student is investing in research.
The class must contain a boolean method milga() that returns true if the student is eligible to receive a grant. The grant criteria are described below.

The grant criteria, as defined by the institution management, are as follows:

- **Bachelor's degree:** At least 10 courses with an average greater than 95.
- **Master's degree:** At least 7 courses with an average greater than 90, and writing a thesis.
- **Doctorate:** At least 2 courses and more than 25 weekly hours invested in research.

Implement for all the classes all the relevant and necessary methods (including getters and setters).

Use the following main program to test your code:

```
#include"MA.h"
#include"PHD.h"


int main()
{
	BA ba[2];
	cout << "Enter details for 2 BA students\n";
	for (int i = 0; i < 2; i++)
		ba[i].input();
	cout << "BA students entitled for milga are: \n";
	for (int i = 0; i < 2; i++)
	{
		if (ba[i].milga())
```

```
                ba[i].print();
            //else cout << " Student not entitled for milga\n";
        }

        cout << "Enter details for 2 MA students\n";
        MA ma[2];
        for (int i = 0; i < 2; i++)
            ma[i].input();
        cout << "MA students entitled for milga are: \n";
        for (int i = 0; i < 2; i++)
        {
            if (ma[i].milga())
                ma[i].print();
        }

        cout << "Enter details for 3 PHD students\n";
        PHD phd[3];
        for (int i = 0; i < 3; i++)
            phd[i].input();
        cout << "12PHD students entitled for milga are: \n";
        for (int i = 0; i < 3; i++)
        {
            if (phd[i].milga())
                phd[i].print();
        }

        return 0;
}
```

The following example yields the output in the submission box:

Add the following 7 students:

BA

(2 students)

Student Id	Surname	First Name	No. Courses	Grades
123	avraham	abe	3	90, 100, 90
234	avrahamson	yitzhak	10	100, 100, 90, 100, 90, 100, 90, 100, 90, 100

MA

(2 students)

Student Id	Surname	First Name	No. Courses	Grades	Thesis
345	jacobson	yaakov	7	90, 100, 90, 100, 90, 100, 90	0
456	emanu	sara	7	90, 100, 90, 100, 90, 100, 90	1

PHD

(3 students)

Student Id	Surname	First Name	No. Courses	Hours in Research
567	imanu	rivka	1	30
678	jacobs	rachel	2	20
789	jacobs	leah	2	30

For the given students the expected output is:

```
ID: 234
first name : yitzhak
last name: avrahamson
num courses: 10
grades: 100 100 90 100 90 100 90 100 90 100
ID: 456
first name : sara
last name: emanu
num courses: 7
grades: 90 100 90 100 90 100 90
research: yes
ID: 789
first name : leah
last name: jacobs
num courses: 2
hours: 30
```

Good Luck!