CENG 301, Spring 2024
Homework Assignment 1

Due: 23:59, March 18, 2024

In this homework, you will implement two data structures.

PART 1 (40 Points): Imaginary Point Class

In this part of the assignment, you will implement the `ImaginaryPoint` class, which represents imaginary numbers in the form of $a+bi$, where i represents the imaginary unit, $\sqrt{-1}$. The class includes functionality for initializing imaginary points, computing their magnitude squared, accessing their real and imaginary parts, overloading arithmetic operators for addition, subtraction, and multiplication, and defining the less than (`<`) operator based on the comparison of their magnitude squared.
The class must adhere to specific naming and interface requirements, as outlined in the provided header file (`ImaginaryPoint.h`) and its corresponding implementation file (`ImaginaryPoint.cpp`). Below, you can find the header file `ImaginaryPoint.h`, we expect you to implement `ImaginaryPoint.cpp`

```
// DO NOT MODIFY THIS FILE
#ifndef IMAGINARYPOINT_H
#define IMAGINARYPOINT_H
class ImaginaryPoint{
private:
    int real_part; // a
    int imaginary_part;//b
public:
    ImaginaryPoint(int real, int imaginary);
    ~ImaginaryPoint();

    int get_magnitude_squared() const; // a*a + b*b

    int get_real_part() const;
    int get_imaginary_part() const;

    // Overload operators
    ImaginaryPoint operator+(const ImaginaryPoint& other) const;
    ImaginaryPoint operator-(const ImaginaryPoint& other) const;
    ImaginaryPoint operator*(const ImaginaryPoint& other) const;
    bool operator<(const ImaginaryPoint& other) const;
};
#endif
```

PART 2 (60 Points): Bottom K List

Bottom K List is a data structure that stores pointers to only the smallest k of the added elements in an array in sorted order.
The data structure will have the following two functionalities:

1. Add a new element.
2. Retrieve the ath smallest element where $0 \leq a < k$, remembering that the 0th smallest element is actually the smallest one (this is known as zero-indexing)

When we put a new element into this data structure, we make sure to place it in the right spot so that all the numbers stay in order from smallest to largest. For example, if we have the numbers 1, 3, and 5 in the box, and we want to add the number 4, we'll put it in between 3 and 5 because it's greater than 3 but less than 5. To illustrate it, we put some comments in the "example test code".
The class implementation must adhere to specific naming and interface requirements, as outlined in the provided header file (BottomKList.h). Below, you will find the header file BottomKList.h, where you are expected to implement the functions by modifying the sections of code indicated by comments.

```
#ifndef BOTTOMKLIST_H
#define BOTTOMKLIST_H
template<typename T, int k>
class BottomKList {
private:
    T* arr[k];
public:
    BottomKList()
    {
        // MODIFY HERE: Implement the constructor.
    }
    ~BottomKList()
    {
        // MODIFY HERE: Implement the destructor (if needed).
    }

    void addElement( T* element )
    {
       // MODIFY HERE: Implement the addElement function.
    }
    T* getBottomXthElement ( int x ) const
    {
        // MODIFY HERE: Implement the getBottomXthElement function.
        // do not forget to return nullptr if x >= k or x < 0
    }
};

#endif
```

Example test code:

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

#include "BottomKList.h"
#include "ImaginaryPoint.h"

int main() {
    BottomKList<int,3> l1;
    int *a = new int(1),
        *b = new int(0),
        *c = new int(5),
        *d = new int(3),
        *e = new int(7),
        *f = new int(-1);
    l1.addElement(a);
```

```
    // now l1.arr is [a, nullptr, nullptr]
    l1.addElement(b);
    // now l1.arr is [b, a, nullptr]
    l1.addElement(c);
    // now l1.arr is [b, a, c]
    l1.addElement(d);
    // now l1.arr is [b, a, d]
    l1.addElement(e);
    // now l1.arr is [b, a, d]
    l1.addElement(f);
    // now l1.arr is [f, b, a]
    int* l1_zero = l1.getBottomXthElement(0);
    int* l1_one = l1.getBottomXthElement(1);
    int* l1_two = l1.getBottomXthElement(2);

    cout<<(*l1_zero)<<endl;
    cout<<(*l1_one)<<endl;
    cout<<(*l1_two)<<endl;

    BottomKList<ImaginaryPoint,2> l2;

    ImaginaryPoint *i1 = new ImaginaryPoint(3,4);
    ImaginaryPoint *i2 = new ImaginaryPoint(5,12);
    ImaginaryPoint *i3 = new ImaginaryPoint(1,1);
    ImaginaryPoint *i4 = new ImaginaryPoint(2,4);
    ImaginaryPoint *i5 = new ImaginaryPoint(20,35);

     // now l2.arr is [nullptr, nullptr]
    l2.addElement(i1);
    // now l2.arr is [i1, nullptr]
    l2.addElement(i2);
    // now l2.arr is [i1, i2]
    l2.addElement(i3);
    // now l2.arr is [i3, i1]
    l2.addElement(i4);
    // now l2.arr is [i3, i4]
    l2.addElement(i5);
    // now l2.arr is [i3, i4]

    ImaginaryPoint* l2_zero = l2.getBottomXthElement(0);
    ImaginaryPoint* l2_one = l2.getBottomXthElement(1);

    cout<<(l2_zero->get_real_part())<<endl;
    cout<<(l2_zero->get_imaginary_part())<<endl;
    cout<<(l2_one->get_real_part())<<endl;
    cout<<(l2_one->get_imaginary_part())<<endl;

    return 0;
}
```

Output of the example test code:

```
-1
0
1
1
```

```
1
2
4
```

IMPORTANT NOTES:
Do not start your homework before reading these notes!!!

NOTES ABOUT IMPLEMENTATION:

1. You ARE NOT ALLOWED to use STL.
2. Output message for each operation MUST exactly match the format shown in the output of the example code.

NOTES ABOUT SUBMISSION:

1. In this assignment, you must have separate interface and implementation files (i.e., separate `.h` and `.cpp` files) for the Imaginary Point class. For the BottomKList class, you will implement it in the header file. Your class names MUST BE `ImaginaryPoint` and `BottomKList`, and your file names MUST BE `ImaginaryPoint.h`, `ImaginaryPoint.cpp`, and `BottomKList.h`. Note that you shouldn't write additional classes in your solution.
2. This assignment is due by 23:59 on Monday, March 18, 2024. You should upload your work to ODTUClass before the deadline. Submissions will be done through VPL. No hardcopy submission is needed.
3. This homework will be graded by your TA **Burak Ferit Aktan** (aktan@ceng.metu.edu.tr). Thus, you may ask your homework related questions directly to him. There will also be a forum on ODTUClass for questions.