Data Structures (Spring 2020)
Lab #3: Object Oriented Programming

Task-1: Classes, Objects, Constructors, Destructor

Create a C++ class called CTime with following members:

- **Attributes** (member variables):
    1. hours: int
    2. minutes: int
- **Constructors**:
    1. **Default (No Argument) Constructor**: This constructor should initialize the hours and minutes with 0. The constructor should also print a message “Default Constructor” followed by the values of hours and minutes.
    2. **Two Argument Constructor**: The construct should take 2 arguments (hours and minutes) and initialize the class attributes hours and minutes with these arguments. The constructor should also print a message “2-arg Constructor” followed by the value of hours and minutes.
- **Destructor:** Create a destructor for the class which prints a message “Destructor” followed by the values of hours and minutes.
- Write a **main** function to test the proper functionality of your class and its members.

```
int main()
{
    cout << "Start of the main function:"<<endl;

    CTime t1;
    CTime t2(2, 25);

    cout<<"End of the main function:"<<endl ; ;
    return 0;
```

Expected output of your main function is similar to the following.

```
Start of the main function:
Default Constructor 0:0
2-args Constructor 2:25
End of the main function:
Destructor 2:25
Destructor 0:0
```

When you run your program:

- Observe the order in whic the constructors and destructors are called.
- Note that the objects are destroyed in the reverse order in which they have been created.

Task 2: Setter/Getters methods

create following member methods in your class and their proper functionality.

1. **inputTime():** A member function that askes the user to enter the hours and minutes and stores the values entered by the user in the attributes.
2. **printTime():** A member function that displays the attributes in the format XXh:YYm, where XX and YY are the values of attributes hours and minutes respectively, Call this function from the main.
3. **setHours():** A member function that takes the value of hours as an argument and stores it in the attribute hours.
4. **getHours():** A member function that returns the value of attribute hours to the calling program.
5. **setMinutes():** A member function that takes the value of minutes as an argument and stores it in the attribute minutes.
6. **getMinutes():** A member function that returns the value of attribute minutes.

Task 3: Overloaded Operators:

1. **Binary Plus (+):** Overload the binary plus operator so that it can add two CTime objects.
2. **Binary Minus (-):** Overload the binary minus operator so that it can subtract two CTime objects.
3. **Insertion operator (>>):** Overload the insertion operator such that you can write statemtents like cin>>t1; in your program, where t1 is an object of type CTime. Note that type of cin is istream.
   Hint: You have to overload the operators as a global function.
4. **Extraction operator (<<):** Overload the extraction operator csuch that you can write statements like cout<<t1; in your program where t1 is an object of type CTime. Note that the the type of cout is ostream.

   Add following statements in your program to test the proper functionalities of above-mentioned operators.

```
cin>> t1;

CTime t3 = t1 + t2;

CTime t4 = t2 – t1;

cout<<"Cumulated time: "<<t3;

cout<<”Time difference:”<<t4;
```

Task 4: Array of Objects

- Declare a static (fixed size) array of 5 CTime objects, populate the array elements with proper values and traverse the array. While traversing the array call printTime() method for each object.
- Create a dynamic array of CTime objects. The size of the array should be taken as input from the user. Dynamically allocate the array in Heap memory, populate it with proper values, display its content.

Task 5: Separate Compilation

- To make your program clearer separate your source code into 3 files:
    - ctime.h: contains the declaration of the CTime class
    - ctime.cpp: contains the definitions of the member functions of the class CTime
    - main.cpp: contains instructions that create and use the instances of the class CTime

Additional Task for Practice

Repeat Task 4 and instead of creating static and dynamic arrays of objects, create arrays of "pointer to objects" and test the functionality of your class methods.