Object Oriented Programming-ITU-2023 -

LAB 2

Task 1 Timer Class

Make a class timer with the following ADT:

```
struct Position
{
    int ri, ci;
};
class Timer
{
    int hour, min, sec;
    Position screen_position;
    string zone;
public:
    Timer(int sr, int sc, int h=0, int m=0, int s=0); // sr and sc are screen
coordinates
    void increment(); // it shd increment second by 1, and update min and hour
accordingly
    void printOnScreen();  // This must print both in standard and universal time
};
```

Throw exceptions if the user initializes the timer with invalid hour, minute or second values.

Read from file 5 timers and their positions and display them over the screen, one at each corner and one in the middle. The timer must keep moving after each one second refresh the new timer values.

Task 2 Set Class

Implement the class Set which should mimic the following Set operations over collection of integers:

- **Union**
- **Intersection**
- **Subtraction**
- **Complement**
- **Disjoint:** It will return True/False.Two sets are disjoint if there is no common element in between them.
- **Proper Subset** : It will return True/False. The first set will be treated as the main set. A is a proper subset of B if all the elements of A are in B and B has at least one element which is not in A
- **Superset**: A set A is a superset of another set B if all elements of the set B are elements of the set A. It will return True/False. The second set will be treated as main set and the above function Proper subset can yield the result
- **Subset**: It will return True/False. The first set will be treated as the main set.
- **Equal set**: It will return True/False. The second set will be treated as the main set.
- **Default constructor** that will treated as an empty set
- **Copy constructor** to implement deep copy
- **Destructor** to delete Vs heap memory
- Input from file and Printing
- Appending sets to set array taking input from console
- Store the records in a file

Note that all the elements in the set have to be unique elements (without any duplicates allowed). Fill in the blanks, const is needed or not

```
class Set
{
        int size;  // size of the set
        int * Vs;  // values of the set
        bool isPresent(int *Vs, Size, int T);  // Make this function as a utility function which should tell whether T is in Vs or not.
 public:
        Set()
        Set(int size)
        Set(const Set & s)
        Set(ifstream& reader)
        void Load( ifstream& rdr)
        void init()_____
        void print_set() ______
        void store(ofstream & wtr)______
        Set Intersection(const Set & B) const ______
        Set Union(const Set & B) const ______
        Set Complement(const Set & U) const ______
        Set Subtraction(const Set & B) const ______
        bool IsEqual(const Set & B) const ______
        bool Disjoint(const Set & B) const ______
        bool IsSuperset(const Set & B) const ______
        bool IsSubset(const Set & B) const ______
        bool IsProperSubSet(const Set & B) const ______
        ~Set()
};
```

Sets Union	Sets Intersection	Sets Subtraction
A U B = {1, 2, 3, 5} U {2, 4 ,3} A U B = {1, 2, 3, 4, 5}	A ∩ B = {1, 2, 3, 5} ∩ {2, 4 ,3} A ∩ B = {2,3}	A - B = {1, 2, 3, 5} - {2, 4 ,3} A - B = {1, 5}
Set Complement	**DisjointSets**	
A' = U - A = {1, 2, 3, 4, 5, 6, 7} - {2, 4 ,3} A' = {1, 5, 6, 7 }	A ∩ B = {1,2,3, 4} ∩ {5,6,7,8} A ∩ B = { }	
3 20 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 5 2 3 4 5 6 3 8 4 1		**You may assume the [0] index set is the universal set.**

Your main menu should have all the functionalities mentioned.

- One option should be there to add a new entry in the list of set.