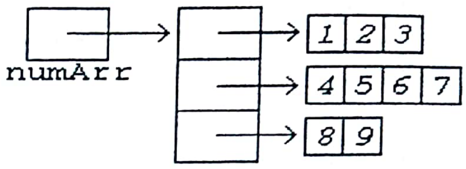
Objective:

- This lab should help in gripping the concept of object manipulation involved while object transition in case of aggregation/composition and array of objects and operator overloading as well and it should also clear many concepts related to dynamic memory allocation for objects.

Challenge-A: *Jagged Array* (13.5)

In computer science a jagged array, also known as a ragged array, is a type of multidimensional array whose elements consist of one-dimensional arrays, hence it is an "array of arrays". The array is called "jagged" because the row/array can have different sizes, producing rows with different number of elements when visualized as output. In contrast, C++-styled arrays are always rectangular (all rows of same sizes).



Layout of the same array

We want to create an ADT named 'JaggedArray', which supports creation of jagged arrays. The operations needed for JaggedArray class and its data representation is given below.
To implement JaggedArray, we have used class Array that we implemented in a practice/lecture; its interface is given at the end of the question so you don't need to recall any of Array class functions. Following is the 'JaggedArray' class, whose detail is given below and also a sample run of JaggedArray class: You are required to implement all the functions given in the class.

```
class JaggedArray
{
     Array * * data;
     int rows;
     bool isValidRow( int r ) const;
public:
     JaggedArray();
     JaggedArray(int r, ...);
     JaggedArray( const JaggedArray & ref );
     Array & operator [] (int i);
     const Array & operator [] (int i) const;
    int getRows() const;
     int getColumns(int r) const;
     ~JaggedArray();
};
```

Jagged Array Data Representation

- data
  'data' will point to an array of pointers of type 'Array' whose each location will point to an Array object.
- rows
  It contains the number of rows of jagged array (number of array objects / size of array pointed by 'data').

Jagged Array Operations

- `bool isValidRow( int r) const;` (0.5)
  return true if the index received is a valid row number otherwise returns false.

- `JaggedArray(int r=0, ...);` (2.0)
  Its first argument 'r' receives the number of rows of jagged array and variable argument list receives the size of each array/row in jagged array.
  For Example if we create object like:
  JaggedArray ja(3,3,4,2);
  Then, it creates the same shape in memory as given in above diagram.
  If nothing is received in 'r' or invalid(negative) value is received then it initializes data and rows to 0.

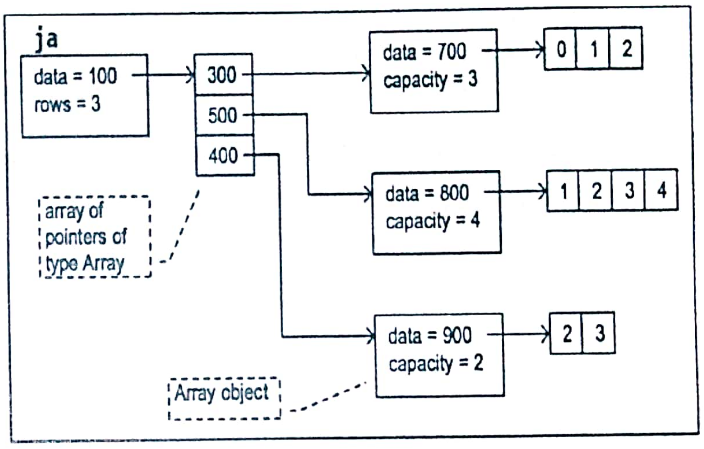
- `JaggedArray( const JaggedArray & ref );` (4.0)

- `Array & operator [] (int i);` (1.5)
  returns an alias of the Array object pointed by location at data[i]
- `const Array & operator [] (int i) const;` (1.5)
  returns an alias of the Array object pointed by location at data[i]
- `int getRows() const;` (0.5)
  return the number of rows.
- `int getColumns(int i) const;` (1.0)
  return the number of columns at row 'i'
- `~JaggedArray();` (2.5)

Sample Run

According to the sample-run/code given below: the object layout will be as follows:

```
int main()
{
    JaggedArray ja(3,3,4,2);
    for ( int i=0; i<ja.getRows(); i++)
    {
        for (int j=0; j<ja.getColumns(i); j++)
        {
            ja[i][j] = i+j;
        }
    }
    for ( int i=0; i<ja.getRows(); i++)
    {
        for (int j=0; j<ja.getColumns(i); j++)
        {
            cout<<ja[i][j]<<" ";
        }
        cout<<"\n";
    }
    return 0;
}
```



Below is the class 'Array' declaration to recall your memory. You are not allowed to add or remove anything in Array other than what is given below.

```
class Array
{
    int * data;
    int  capacity;
    bool isValidIndex( int index ) const;
public:
    Array(int argCount=0, ...);
    Array (const Array & ref);
    Array & operator = (const Array & ref);
    ~Array();
    int getCapacity() const;
    int & operator [] (int index);
    const int & operator [](int index) const;
    void reSize ( int newCap );
};
```

BSEF18M031

Challenge-B: ***Smart Watch*** (54.5)

Did you know that your body weight is approximately 60 percent water? Your body uses water in all its cells, organs, and tissues to help regulate its temperature and maintain other bodily functions. Because your body loses water through breathing, sweating, and digestion, it's important to rehydrate by drinking fluids and eating foods that contain water. The quantity of water you need depends on a variety of factors, including the climate you live in, how physically active you are, and whether you're experiencing an illness or have any other health problems.

In this task, we shall design a Smart Watch. Actually, a first step towards smart watch ☺. We shall enhance the functionality of class 'Watch' implemented in last lab.

We shall add the water logging and related reporting feature in our watch so the user can see how much water he intake. In order to achieve the said purpose, we need to implement a bunch of classes listed below:

Note:

- It is strongly recommended to complete them in the order they are listed in this document.
- The `Time` and `CString` classes are not part of evaluation of this lab. But they will be evaluated today by the TAs separately and separate marks will be rewarded considering it a surprise quiz.
- The same goes for `Date` class well except the decrement `day`, `month`, `year` functions, which will be evaluated too but their marks will be added in the lab task as well.

Although our main/core class is `SmartWatch` but to complete it, we need to complete many supporting classes which needs to be implemented before we do anything for `SmarWatch`. The detail of each of such classes are given below.

Responsible for storing Date.		**4**

`class Date`
`{`
`    BoundedInteger day;`
`    BoundedInteger month;`
`    BoundedInteger year;`
`    static const int daysInMonth[ 13 ];`
`public:`
`    Date();`
`    Date(int,int,int);`
`    void setDate(int,int,int);`
`    void setDay(int);`
`    void setMonth(int);`
`    void setYear(int);`
`    bool isLeapYear ();`
`    int getDay() const;`
`    int getMonth() const;`
`    int getYear() const;`
`    void printFormat1() const;`
`    void printFormat2() const;`
`    void printFormat3() const;`
`    void incDay(int=1);`
`    void incMonth(int=1);`
`    void incYear(int=1);`
`    CString getDateInFormat1() const;`
`    CString getDateInFormat2() const;`
`    CString getDateInFormat3() const;`
`    void decDay(int=1);`
`    void decMonth(int=1);`
`    void decYear(int=1);`
	You already know about it.	3

`    int isEqual(Date d);`	//if (*this==d) -> 0, if (*this>d) -> 1, if (*this<d) -> -1	1
`};`		
Responsible for storing Time.		**2**

`class Time`
`{`
`private:`
	You already know about it.	2

```
    BoundedInteger hour;
    BoundedInteger minute;
    BoundedInteger second;
public:
    Time ( int = 0, int = 0, int = 0);
    void setMinute ( int m );
    void setSecond ( int s );
    void setHour(int h);
    void setTime ( int h, int m, int s );
    int getHour ( )  const;
    int getMinute ( ) const;
    int getSecond ( ) const;
    void printTwentyFourHourFormat() const;
    void printTwelveHourFormat() const;
    void incSec( int inc= 1 );
    void incMin( int inc= 1 );
    void incHour( int inc= 1 );
    void decSec( int inc= 1 );
    void decMin( int inc= 1 );
    void decHour( int inc= 1 );
};
```

Each Object of following class represents one water log (quantity of water drank at any given date/time). See the data members which are self-explanatory. — 4 — 3.5

`class WaterLog`		
`{`		
`float quantity;`		
`Date waterDate;`		
`Time waterTime;`		
`public:`		
`WaterLog ( float = 0.5 );`	Default water quantity is 0.5. waterDate/waterTime set to current Date and Time.	2 — 2
`WaterLog ( float, Date, Time );`	Initialized the data members with the received values.	1 — 0.5
`void setQuantity ( float );`		
`void setWaterDate ( Date );`		
`void setWaterTime( Time );`		
`float getQuantity( );`		1 — 1
`Date getWaterDate( );`		
`Time getWaterTime( );`		
`};`		

This class is responsible for maintaining the list of water logs. Its object will be composed in smart watch. It is our core component where all the features related to water intake are placed. — **36**

`enum WaterUnits { MILLI_LITRE, LITRE };`	Enumeration for water measuring units.	
`class WaterLogList`		
`{`		
`WaterLog * * watLogArray;`	Array of pointers instead of array of objects to save memory. watLogArray will point to array of pointers, whose each location will point to a WaterLog object on heap or will be null.	
`WaterUnits defaultWaterUnit = MILLI_LITRE;`	setting by default water unit to MILLI_LITRE, which may be changed to LITRE, If so then all previous stored logs quantity will be converted.	
`int waterLogCount;`	Represents the count of water logs. i.e. the number of WaterLog objects in the watLogArray.	
`int waterLogCapacity;`	Represents the size of array pointed by watLogArray.	
`void reSize( );`	Resize the array pointed by watLogArray by doubling the size of it.	
`public:`		
`WaterLogList( );`	Initialize watLogArray to an array of size 10.	2 — 1.5
`~WaterLogList( );`	You know what to do.	3 — 1.5
`void logWater( float );`	Create an object on heap of WaterLog and add it in the array pointed by watLogArray. Set the Date and Time of water logging to current	1 ✗

	date/time Resize the array if it gets full.		
`void logWater( float, Date, Time );`	Create an object on heap of `WaterLog` with received values and add it in the array pointed by `watLogArray`. Resize the array if it gets full.	1	✓
`WaterUnits getDefaultWaterUnit( );`		1	1
`void changeDefaultWaterUnit( WaterUnits );`	If water unit is changed then you also need to change the unit value/quantity in the previous logs.	4	4
`float totalWaterIntakeToday( ) const;`	Returns the total water intake today/current-date.	4	4
`float totalWaterIntakeCurrentMonth( ) const;`	Returns the total water intake in current month.	4	4
`float totalWaterIntakeInLastNDays( int ) const;`	Returns the total water intake in last N days.	4	4
`void printMonthwiseHistogram( int ) const;`	A bar graph to display, which shows monthly intake of water for last N months. See the explanation on the last page.	6	6
`void printDaywiseHistogramLastNDays( int ) const;`	A bar graph to display, which shows daily intake of water for last N days. See the explanation on the last page.	6	6
`};`			
The class which exhibits the behavior of a smart watch by composing `Date`, `Time` and `WaterLogList` objects.		**3.5**	1
`class` **`SmartWatch`**			
`{`			
`Watch wch;`			
`WaterLogList wLogList;`			
`public:`			
`SmartWatch ( );`	Initialize the watch data and time with current date/time.	1.0	0.5
`SmartWatch ( Date, Time );`	Initialize the watch data and time with given date/time.	1.0	0.5
`void setWatch ( Watch );`		0.5	
`Watch & getWatch ( );`		0.5	
`WaterLogList & getWaterLogList ( );`		0.5	
`};`			
Write a sample run / code in main, which creates the object of `SmartWatch` class and exhibits all the features of it. TA will ask you to get certain behavior/information for SmartWatch object.		5	
`int main ( ) { SmartWatch sw; return 0; }`		5	

Explanation for `printDaywiseHistogramLastNDays` and `printMonthwiseHistogram`

Assume, **Today is 12 October, 2019**
And user has stored following data in his smart watch so far:

```
    Water Quantity = .2ml        Date = 12-10-2019       Time = 13:15:50
    Water Quantity = .3ml        Date = 12-10-2019       Time = 8:1:15
    Water Quantity = 1ml         Date = 11-10-2019       Time = 3:10:0
    Water Quantity = 3ml         Date = 11-10-2019       Time = 6:10:12
    Water Quantity = .4ml        Date = 1-8-2019         Time = 18:34:45
    Water Quantity = 1ml         Date = 10-10-2019       Time = 8:20:50
    Water Quantity = 7ml         Date = 4-8-2019         Time = 8:1:15
    Water Quantity = 5ml         Date = 17-11-2017       Time = 12:1:10
    Water Quantity = 4ml         Date = 8-8-2019         Time = 23:11:55
    Water Quantity = 2ml         Date = 12-10-2019       Time = 19:10:39
```

On executing the `printDaywiseHistogramLastNDays(3)`, The following output will be displayed.
Where each * represents `0.25ml`.

```
2019-10-12 : **********:2.5
2019-10-11 : ****************:4
2019-10-10 : ****:1
```

The same style of output will come on executing `printMonthwiseHistogram`. The only difference is: each * represents 0.5ml.
For Example, on execution of `printMonthwiseHistogram(3)`, following output will be displayed.

```
2019-10 -> ***************:7.5
2019-09 -> :0
2019-08 -> ***********************:11.4
```

In a speech Arnold Schwarzenegger said that I became very friendly with Muhammad Ali in the 70s. "Muhammad Ali worked his butt off. I saw it first-hand. I remember there was a sports writer in the gym, and Ali was doing sit-ups then. He asked Ali, how many sit-ups do you do? He said,

I don't start counting until it hurts."

Now think about it, he doesn't count until it hurts, until he feels the pain that is waking hard. So, you can't get around the hard work it doesn't matter who it is. As a maverick, I believe what the Tedd Turner said,

"Work like hell and advertise".

-- Arnold Schwarzenegger --