CC-216: Data Structures & Algorithms
BS SE/CS Fall 2020
Practice - 01

Issue Date: 2-Mar-2022

Objective:

- The practice will help you to dig in the features of templates.

Task – 1:
In class, we developed the function 'void mySwap(T &, T &)' using aliases to swap the contents of the received entities.
Now develop the same function again but using pointers i.e. 'void mySwap(T *, T *)'.

Task – 2:
Write a function template named 'myGenericSort', which receives an array of size N and rearranges the elements of the array in ascending order.
Test your code by passing arrays of different types (primitive and user define).

Task – 3: ***Generic Array Class***
Complete the generic Array class discussed in class today.

```
template< typename T>
class Array
{
    T * data;
    int capacity;
    int isValidIndex( int index ) const;
public:
    ~Array();
    T & operator [] ( int i );
    const T & operator [] ( int i ) const;
    int getCapacity() const;
    void reSize ( int newCapacity );
    Array<T> & operator = ( const Array<T> &);
    Array ( const Array<T> &);
    Array<T> & operator = (Array<T> &&);
    Array (Array<T> &&);
    Array(int=0);
};
```

You are recommended to test your generic Array class with the following code and also with any user define type that you may pick yourself (do it with String class too, developed in OOP).

Sample Run:

Code

```
int main()
{
    Array< Array < Array< int > > > c;
    c.reSize(2);
    int val=1;
    for ( int i=0; i<2; i++)
        c[i].reSize(3);
    for ( int i=0; i<c.getCapacity(); i++ )
    {
        for ( int j=0; j<c[i].getCapacity(); j++ )
        {
            c[i][j].reSize(4);
        }
    }
    for ( int i=0; i<c.getCapacity(); i++ )
    {
        for ( int j=0; j<c[i].getCapacity(); j++ )
        {
            for ( int k=0; k<c[i][j].getCapacity(); k++ )
            {
                c[i][j][k]=val;
                val++;
            }
        }
    }
    for ( int i=0; i<c.getCapacity(); i++ )
    {
        for ( int j=0; j<c[i].getCapacity(); j++ )
        {
```

Console Output

```
1   2   3   4
5   6   7   8
9   10  11  12

13  14  15  16
17  18  19  20
21  22  23  24
```

```
            for ( int k=0; k<c[i][j].getCapacity(); k++ )
            {
                cout << c[i][j][k]<<"\t";
            }
            cout<<"\n";
        }
        cout<<"\n";
    }
    return 0;
}
```

Task – 4: ***Generic Set Class***

The following 'Set' class declaration is designed for the Set of integers. You may have developed it in your OOP class.
In this task, you are required to convert the following class into Generic 'Set' class/Class-Template.

```
class Set
{
    int * data;
    int capacity;
    int noOfElements;
public:
    Set( int cap = 0 );
    ~Set();
    Set(const Set &);
    Set(Set &&);
    Set & operator = (const Set &);
    Set & operator = (Set &);
    void insert (int element);
    void remove (int element);
```

```
    void print() const;
    int getCardinality() const;
    int getCapacity() const;
    int isMember ( int val ) const;
    int isSubSet ( const Set & s2 ) const;
    void reSize ( int newcapacity );
    Set calcUnion ( const Set & s2 )
const;
    Set calcIntersection ( const Set & s2
) const;
    Set calcDifference ( const Set & s2 )
const;
    Set calcSymmetricDifference ( const
Set & s2 ) const;
};
```

Sample Run for a non-template Set class:

Code

```
int main()
{   Set<int> s1, s2(3);
    ~~Set s1, s2(3);~~
    s1.insert(5);
    s1.insert(34);
    s1.insert(7);
    s1.insert(54);
    s1.insert(78);
    s1.insert(5);
    s1.insert(45);

    s2.insert(2);
    s2.insert(45);
    s2.insert(7);
    s2.insert(6);

    cout<<"\nSet s1 = ";
    s1.print();

    cout<<"\nSet s2 = ";
    s2.print();

    cout<<"\ns1 ∩ s2 = ";
    s1.calcIntersection(s2).print();

    return 0;
}
```

Console Output

```
Set s1 = {5, 34, 7, 54, 78, 45}
Set s2 = {2, 45, 7, 6}
s1 ∩ s2 = {7, 45}
```

Don't forget to thoroughly test each and every function with different primitive and user define types.