Assignment 1 - Practice 6 – OOP
Dynamic Memory Allocation

Note: The viva will be held in next lab.

You are required to create following classes/ ADT's:

1. Rational
2. Set
3. Bag
4. Complex Number
5. Matrix
6. MyString

For each of these classes, you have to write following member functions:

- parameterized constructor with default value of parameters
- copy constructor, assignment operator and destructor; wherever required
- stream insertion and extraction operator
- +, += operator
- *, *= operator for matrix, rational & complex number class
- For **Rational** class, write functions:
    - to type cast object in float (see example at the end of this document)
- For **set** class, write functions:
    - to find union of two sets in a new set
    - to find intersection of two sets in a new set
    - the difference of two sets in a new set
    - to check, whether current object is subset or not of second object
- For **matrix** class, write functions:
    - to check, whether matrix is in Echleon form
    - to check, whether matrix is in Reduced Echleon form
    - function to perform row operations, like rowOp(2, 4, 3, 1) means, multiply row 4 with 2, row 1 with 3, add them and store result in row 1. For subtraction, negative parameters can be used
- For **MyString** class, write functions:
    - to find sub-string and return index or -1
    - to compare two strings with case
    - to compare two strings without case
    - to return an array of strings having words (the object may have multiple words) terminated by null string
    - to type cast object into integer
    - to type case object into float
    - to replace substring with another substring (may both substrings have different size)
-

Write separate class files for each. In each class definition, declare the large functions and provide their implementation below the class in the same file. Finally, write driver programs (with main functions) to test each class. Here is an explanation of each class.

1. Rational:

This class is to represent an object of rational number having integer numerator and denominator. The denominator cannot be zero. Print object in fraction, by caring of negative signs appropriately.

2. Set:

List of integers without duplication. The size is extendible, means user can add any number of elements. The "+" operator with integer will create a new object having elements of current object plus new integer (if not

repeating). The "+=" operator with integer will add new integer in current object (if not repeating). The "+" operator with set object will create a new object having unique elements of both sets. The "+=" operator with set object will add unique elements of second set in the current object. The stream operator will print current size of set in first line and all elements in next line separated with spaces.

3. Bag:

Bag can have duplicate entries. Rest of the functions are similar to Set class.

4. Complex Number:

Complex number use to represent numbers having under root of negative one. The object has two parts: one is real and the other is imaginary, both of which can be zero. When printing the object, ensure the signs are considered, and only the non-zero part should be printed.

5. Matrix:

The object has three data members: rows, columns and elements (two-dimensional dynamic array). Write private function to initialize elements randomly in small range of integers. For addition and multiplication, check whether the operation is possible or not? In case of "+=" and "*=" operations do nothing. In case of "+" and "*" operation return a matrix with zero rows and zero columns. Print matrix object in row, column form without the information of rows and columns.

6. MyString:

The object has two data members, size and character array to store the string.

Here, is example of type casting operator:

```
class ABC{
            int a, b, c;
      public:
            ABC(){
                  a = rand() % 10;
                  b = rand() % 10;
                  c = rand() % 10;
            }
            //This operator has no return type but it definitely return as per its
type
            operator int(){
                  return a + b + c;
            }
            friend ostream& operator << (ostream &out, const ABC &abc){
                  out << abc.a << ' '  << abc.b << ' '  << abc.c << '\n';
                  return out;
            }
};
int main(){
      ABC obj;
      cout << obj;
      cout << (int)obj;
      return 0;
}
```