

Assignment # 5 - On Basics of CLASSES

A. Implement the class **COMPLEX_NUMBERS** which must perform the following tasks between the complex numbers

- Initialization from the file array of complex numbers
- Performing the following operations of complex numbers

• Addition // Already done in class • Subtraction // Already done in class • Multiplication // Already done in class • Finding Multiplicative Inverse // google it.	• Finding Additive Inverse // google the definition. • Division // google it. • Modulus // google it. • Conjugate // google it.
--	---

B. Implement the class **FRACTION** which must perform the following tasks between various fractions

- Initialization from the file array of fractions (numerator and denominator)
- Performing the following operations of complex numbers
- Addition: // **adds two fractions and return the resulting fraction**
 fraction add(**fraction** f1 , **fraction** f2); // Outside Class
 fraction add(**fraction** f2); // Inside Class
- Subtraction: // **subtracts two fractions and return the resulting fraction**
 fraction subtraction(**fraction** f2);
 fraction subtraction(**fraction** f1 , **fraction** f2);
- Multiplication: // **multiply two fractions and return the resulting fraction**
 fraction mul(**fraction** f2);
 fraction mul(**fraction** f1 , **fraction** f2);
- Finding Multiplicative Inverse : // **calculate multiplicative inverse of a fraction and return resulting fraction**
 fraction multiplicative_Inverse();
 fraction multiplicative_Inverse(**fraction** f);
- Finding Additive Inverse: // **calculate additive inverse of a fraction and return resulting fraction**
 fraction Additive_Inverse();
 fraction Additive_Inverse(**fraction** f);
- Division : // **divides fraction and return resulting fraction**
 fraction divide(**fraction** f2);
 fraction divide(**fraction** f1, **fraction** f2);
- **Reduced Fraction** (by dividing both numbers by its **GCD**: and that **GCD** function must be a utility function i.e write **GCD** in **private** part of the class, also make a function **LCM**).
- Printing the Fraction
 - In A/B form
 - In A/B form (in the most reduced fraction)
 - In Floating value form.

C - Implement the class String which must perform the following tasks between various strings.

```
class Mystring
{
    int size;
    char *ptr;

Public:
    // Provide necessary functionality
    // prints the string on console
    void print();
};
```

1. Every possible constructors (with DAM strings) and one destructor
 - Empty , char* , integer (which will call itoa)
2. MyString* Split(int)
3. String equality // test whether two strings are equal or not
 - bool equal(const Mystring & s2) const;
 - int CMP(const Mystring & s2); // just like str_cmp(return 1 if the first string is greater than the second otherwise -1, if equal it should return 0)

```
s1="abcde"
s2="abcde"
cout<<s1.equal(s2)
// evaluates true
```

```
s1="abcde"
s1="1234"
cout<<s1.equal(s2)
// evaluates false
```

```
s1="abcde"
s2="zzzz"
cout<<s1.cmp(s2) ;
// should return -1
```

4. Shallow-Copy
 - First make a function ReplaceFirst(char c) which must replace the first character of the string by c.
5. Copy constructor (deep copy)
6. Tokenization
 - (it returns a string array split based on delimiters)

7. Trim (remove spaces in the beginning and end)
8. String All-substring searching: return an int* and count (taken & by reference) containing all the indexes of the substring and # of times substring found.

9. String concatenation //adds two strings and returns resultant string
 - string str_cat(string s1,string s2);
10. Index // returns character at mentioned position
 - char index (int i);

```
Mystring s1="usama" , s2="is great"
s3=s1.str_cat(s2);
// s3 contains "usamais great"
```

```
Mystring s1="abcde"
char ch= s1.index(2); // ch contains c;
```

```
Mystring s1="abcde"
char ch= s1.index(9);
// ch contains NULL character ;
```

11. Find
 - First // returns first index at which character is found
 - int find_first(char ch);
 - Last // returns last index at which character is found
 - int find_last(char ch);
 - All // returns pointer to an array which contains all indices at which character is present
 - int* find_all(char ch, int &C);

```
Mystring s1="zabcdeaaa"; int C;
int first= s1.find_first('a'); //first=1
int last=s1.find_last('a'); // last=8
int *indices= s1.find_all('a', C); // indices contain {1,6,7,8} and C = 4
```

12. Remove character
 - First // remove only first occurrence of character
 - void remove_first(char ch);
 - Last // remove only last occurrence of character
 - void remove_last(char ch);
 - All // remove all occurrences of character
 - void remove_all(char ch);

```
Mystring s1="sarfraz"
s1.remove_first('a');
// s1 becomes "srfrz"
```

```
Mystring s1="sarfraz"
s1.remove_last('a');
// s1 becomes "sarfrz"
```

```
Mystring s1="sarfraz"
s1.remove_all('a');
// s1 becomes "srfrz"
```

13. Remove At an index // removes the character at given index
 - void remove_at(int i);
14. Add a character at // insert a character a given position
 - void insert_at(int i, char ch);

For example :

For example:

```
Mystring s1="sarfraz" ;
s1.remove_at(3) ;
//s1 becomes "sarraz"
```

```
Mystring s1="uama" ;
s1.insert_at(1,'s');
//s1 becomes "usama"
```

15. Add a string at // insert a substring at given position

- void insert_at(int i,Mystring sub);

16. Clear //clears the string

- void clear();

17. Length // calculates the length of a string

- int length();

For example:
Mystring s1=" is GREAT"
Mystring s2="Usama"
s1.insert_at(0,s2);
// s1 becomes "Usama is GREAT"

For example:
Mystring s1="Sarfraz"
s1.clear()
//s1 becomes ""

For example:
Mystring s1="sarfraz"
int len=s1.length(); // len contains 7

18. Two Static Function in String class:

- Atoi (alphabet to integer conversion) // convert string to integer and Itoa (Integer to Alphabet)

- int Atoi();

- Mystring Itoa(int num);

For example:
Mystring s1="99"
int val= Mystring::Atoi(s1); // Static function
// val+1 is 100

For example:
Mystring val= string::Atoi(123); // Static function
// val+1 is 100

```
class MyString
{
    char* ptr;
    int Size;
Public:
    MyString();
    MyString(const char* s);
    MyString(int);
    MyString(const MyString & MS);
    MyString* Split(char d, int &Count);
    MyString* Tokenize(char* ds, int &Count);
    bool Equal(const MyString &)const;
    int Cmp(const MyString &)const;
    void SetString(const char* p);
    void Trim();
    int* SubStringSearch(const char* substr, int & Count) const;
    int* SubStringSearch(const MyString &substr, int & Count) const;
    MyString Concat(const MyString& S2) const;
    void Print();

    // Add further functions wherever needed.

    ~MyString();
};
```