CS 103 Computer Programming

Assignment Number 4

April 1, 2017

Deadline: Monday 3rd April, 2017 before 21h30

Attention

- Make sure that you read and understand each and every instruction. If you have any questions or comments you are encouraged to discuss your problems with your colleagues (and instructors) on Piazza.
- Plagiarism is strongly forbidden and will be very strongly punished. If we find that you have copied from someone else or someone else has copied from you (with or without your knowledge) both of you will be punished. You will be awarded straight zero in this assignment or all assignments.
- Submit three files "*.h, *.cpp and main.cpp" file for each question of your assignment.

Q1: Operator Overloading for String Class Your goal is to overload the operators for "String" class that you have implemented in the last assignment. You will need to write three files (string.h, string.cpp and stringMain.cpp). Your implemented class must fully provide the definitions of following class (interface) functions . Please also write down the test code to drive your class implementation. **Please note that we will be running your code against our test code and any segmentation faults or incorrect result will result in loss of marks.**

```
#include<iostream>
using namespace std;

class String {
// think about the private data members...
public:
// provide definitions of following functions...
        String(); // default constructor
        String(char *str); // initializes the string with constant cstring
        String(const String &); // copy constructor to initialize the string
         ↪ fromexisting string
        String(int x); // initializes a string of pre-defined size
        char &operator[](int i); // returns the character at index [x]
        char &operator[](int i)const; // returns the character at index [x]

        String operator+(const String &str); // append a String at the end of string
        String operator+(const char &str); // append a char at the end of string
        String operator+(char *&str); // append a String at the end of string

        String operator-(const String &substr); //removes the substr from the string
        String operator-(const string &substr); //removes the substr from the string


        bool operator!(); // returns true if string is empty..

        String& operator=(const String&); // Copy one string to another ...
        String& operator=(char*); // Copy one string to another ...
        String& operator=(const string&); // Copy one string to another ...


        bool operator==(const String&)const; //returns true if two strings are equal
        bool operator==(const string&)const; //returns true if two strings are equal
        bool operator==(char *)const; //returns true if two strings are equal

```

```
34
35          int operator()(char); // returns the index of character being searched.
36          int operator()(const String&); // returns the index of character being
             ↪ searched.
37          int operator()(const string&); // returns the index of character being
             ↪ searched.
38          int operator()(char *); // returns the index of character being searched.
39
40          String operator*(int a); //multiples the string by i times and return the
             ↪ string. Remember the Python functionality for *
41          int length(); // returns the length of string
42          ~String(); // destructor...
43  };
44  ostream& operator<<(ostream& input, const String&); //Outputs the string
45  istream& operator>>(istream& ouput,  String&); //Inputs the string
```

Q2: Implementation of Array Class Your goal is to overload the operators for "Array" class that you have implemented in the last assignment. You will need to write three files (array.h, array.cpp and arrayMain.cpp). Your implemented class must fully provide the definitions of following class (interface) functions . Please also write down the test code to drive your class implementation. **Please note that we will be running your code against our test code and any segmentation faults or incorrect result will result in loss of marks.**

```
1   #include<iostream>
2   using namespace std;
3
4   class Array {
5   // think about the private data members...
6   public:
7   // provide definitions of following functions...
8           Array(); // a default constructor
9           Array(int size); // a parametrized constructor initializing an Array of
             ↪ predefined size
10          Array(int *arr, int size); // initializes the Array with an existing Array
11          Array(const Array &); // copy constructor
12          int& operator[](int i); // returns the integer at index [i] after checking
             ↪ the out of range error
13          int& operator[](int i)const;
14          const Array & operator=(const Array&); //copy the array
15          Array operator+(const Array&); //adds two Array
16          Array operator-(const Array&); //subtracts two Array
17          Array operator++(); //adds one to each element of Array
18          Array operator++(int); //adds one to each element of Array
19          Array& operator--(int); //subtracts one from each element of array
20          bool operator==(const Array&)const; //returns true if two arrays are same
21          bool operator!(); // returns true if the Array is empty
22          void operator+=(const Array&); //adds two Array
23          void operator-=(const Array&); //subtracts two Array
24          int operator()(int idx, int val); // erases the value val at idx. Returns 1
             ↪ for a successful deletion and -1 if idx does not exists or is invalid.
             ↪ Shift the elements after idx to the left.
25          ~Array(); // destructor...
26  };
27  ostream& operator<<(ostream& input, const Array&); //Inputs the Array
28  istream& operator>>(istream& ouput, Array&); //Outputs the Array
```

Q3: Implementation of Matrix Class Your goal is to overload the operators for "Matrix" class that you have implemented in the last assignment. You will need to write three files (matrix.h, matrix.cpp and matrixMain.cpp). Your implemented class must fully provide the definitions of following class (interface) functions . Please also write down the test code to drive your class implementation. **Please note that we will be running your code against**

our test code and any segmentation faults or incorrect result will result in loss of marks.

```
#include<iostream>
using namespace std;

class Matrix {
// think about the private data members...
// the matrix should store real numbers
public:
//include all the necessary checks before performing the operations in the functions
        Matrix(); // a default constructor
        Matrix(int, int); // a parametrized constructor
        Matrix(const Matrix &); // copy constructor
        float &operator()(int &i, int &j); //set value at (i,j)
        float &operator()(int &i, int &j)const; //set value at (i,j)
        Matrix& operator=(const Matrix &); //assigns (copies) a Matrix. Returns the
         ↪ same
        bool operator==(const Matrix &); //Compares two matrices
        Matrix operator+(const Matrix &); //adds two Matrices and returns the result
        Matrix operator-(const Matrix &); //subtracts two Matrices and returns the
         ↪ result
        Matrix operator*(const Matrix &); //multiplies two Matrices
        Matrix& operator++(int); //add one to every element
        void operator+=(const Matrix&); //adds two Matrices
        void operator-=(const Matrix&); //subtracts two Matrices
        ~Matrix();
};
ostream& operator<<(ostream& input, const Matrix&); //Outputs the Polynomial
istream& operator>>(istream& ouput, Matrix&); //Inputs the Polynomial
```

Q4: Implementation of Polynomial Class Your goal is to overload the operators for a generic "Polynomial" class. A polynomial will be represented via its coefficients. Here is a degree of third degree polynomial $4x^3 + 3x + 2$; here their will be four coefficients and the coefficient corresponding to 2nd power is zero. You will need to write three files (polynomial.h, polynomial.cpp and polynomialMain.cpp). Your implemented class must fully provide the definitions of following class (interface) functions . Please also write down the test code to drive your class implementation. **Please note that we will be running your code against our test code and any segmentation faults or incorrect result will result in loss of marks.**

```
#include<iostream>
using namespace std;

class Polynomial {
// think about the private data members...
// the matrix should store real numbers
public:
//include all the necessary checks before performing the operations in the functions
        Polynomial(); // a default constructor
        Polynomial(int); // a parametrized constructor, received the highest degree
         ↪ of polynomial
        Polynomial(const Polynomial &); // copy constructor
        Polynomial& operator=(const Polynomial &); //assigns (copies) a Polynomial.
         ↪ Returns the same
        bool operator==(const Polynomial &); //Compare and return true if equal
        Polynomial operator+(const Polynomial &); //adds two Polynomial and returns
         ↪ the result
        Polynomial operator-(const Polynomial &); //subtracts two Polynomial and
         ↪ returns the result
        void operator+=(const Polynomial&); //adds two polynomials
        void operator-=(const Polynomial&); //subtracts two Polynomials
```

```
18          ~Polynomial();
19  };
20  ostream& operator<<(ostream& input, const Polynomial&); //Outputs the Polynomial
21  istream& operator>>(istream& ouput, Polynomial&); //Inputs the Polynomial
```

Q5: Implementation of Bouquet of Flowers Your goal here is to write classes for creating a bouquet of flowers. To create the bouquet of flower your will need to write following two classes.

Design a class *Flower*. A "flower" is characterized by following attributes:

- a name
- a color
- a basic price per unit
- an indication whether the flower is perfumed or not
- and an indication to know whether the flower is on sale.

and with following behavior:

- a constructor initializing the attributes using parameters given in the order shown by the provided main(); a default constructor will not be necessary but the last two parameters will have false as default value;
- a price method returning the flower's price : the price will be the base price if the flower is not on sale; otherwise, the price will be half the base price;
- a bool parfume() method indicating whether the flower is perfumed or not;
- Overloaded stream insertion operator. The characteristics have to be displayed in strict accordance with the following format :

```
<Name> <Color> <Parfumed>, Price: <Price> Rs.
```

- an overloading of the == operator returning true if two flowers are identical, false otherwise. Two flowers are considered identical if they have the same name, the same color, and the two flowers are both either perfumed or not (neither the price nor the fact that the flower is on sale or not is involved in the comparison).

Next write a "Bouquet" class which will be modeled using a dynamic array of Flowers.
The Bouquet class offers the following methods :

- a method bool parfume() returning true if the bouquet is perfumed and false otherwise; a bouquet is perfumed if at least one of its flowers is perfumed;
- a method price without parameters returning the price of the bouquet of flowers; This is the sum of the prices of all its flowers; this sum is multiplied by two if the bouquet is perfumed;
- a stream insertion method, should display all information of bouquet with the total price. This method will display the characteristics of the bouquet of flowers respecting rigorously the following format :

  If the bouquet does not contain any flower,

```
Still no flower in the bouquet
```

  or :

```
Parfumed Bouquet composed of:
  <Flower1>
    ..
  <FlowerN>
Total Price: <Price_of_bouquet> Rs.
```

  Here $< FlowerX >$ means display of the X_{th} flower of the bouquet in the format specified by the overload of the << operator. There is a newline after displaying each flower and after displaying the price of the bouquet.

- an overload of the += operator which allows adding a flower to the bouquet, the flower will always be added at the end.

- an overload of the -= operator taking as a parameter a flower and removing from the bouquet all the flowers identical to the latter (according to the definition of the == operator);
- an overloaded + operator according its usage in the provided main
- an overloaded - operator according to its usage in the provided main

```
int main() {
        // example of  Yellow oderless rose.
        Flower r1("Rose", "Yellow", 1.5);
        cout << r1 << endl;
        // example of  Yellow parfumed rose
        Flower r2("Rose", "Yellow", 3.0, true);
        // example of  parfumed Red rose on sale
        Flower r3("Rose", "Red", 2.0, true, true);
        Bouquet b1;
        b1 += r1; // add one Flower of r1 type
        b1 += r1; // add another Flower of r1
        b1 += r2;
        b1 += r3;
        cout << b1 << endl;

        b1 = b1 - r1; // Delete all the Flowers of type r1
        cout << b1 << endl;

        Bouquet b2;
        b2 = b1 + r1; // Add one Flower of type r1
        cout << b2 << endl;

        // Delete all the parfumed flowers from the bouquet.
        b2 -= r2;
        b2 -= r3;
        cout << b2;
        return 0;
}
```