CS-1004 Object Oriented Programming Spring-2023
ASSIGNMENT-03
Section (All)

Submission Deadline: 6th April, 2023 at 12:30 PM.
Instructions:

1. Do not use any String or math libraries (such as cmath, cstring, string etc) and also do not use built-in functions (such as strlen, strcmp etc) unless specified. **Caution**: zero marks may be awarded.
2. Do not edit **Function Prototypes**. **Caution**: zero marks may be awarded.
3. The usage of string is strictly prohibited.
4. Your code must be **generic , use dynamically created arrays only**.
5. Correct and timely submission of the assignment is the responsibility of every student.
6. **Evaluation**: The assignment is of 200 marks. All submissions will be evaluated on test cases similar to the ones provided. Failure of a test case would result in zero marks for that part.
7. Marks distribution and test Cases are provided for each question. Your code will be evaluated with **similar test cases**. If the required output is generated, you will be awarded full marks. Failing to generate the correct output may result in zero marks.
8. **Plagiarism**: Plagiarism of any kind (copying from others, copying from the internet, etc) is not allowed. If found plagiarized, you **WILL** be awarded zero marks in the assignment. Repeating such an act can lead to strict disciplinary actions and failure in the course. **8. Please start early otherwise you will struggle with the assignment.**
9. **Test cases:** Test cases (in gtest) will be shared with you on Google Classroom. **We will be running your code against our test cases, and a test case failure or a segmentation fault/incorrect result or even syntax error will result in zero marks.**
10. **Submission Guidelines**: Dear students, we will be using **auto-grading tools (gtest)**, so failure to submit according to the below format would result in zero marks in the relevant evaluation instrument.
    A. Make a folder titled **21X-XXXX_A2_OOP** and put all your .cpp files and .h files in it. Compress the folder as a zip file and upload on **google form** only.
    B. Submission: Header (.h) files are provided where required. Create a .cpp file for each question, name the .cpp file the same as the header. E.g Car.h and Car.cpp.
    C. You need to submit both header file and cpp file for each question.
    D. All cpp files must contain your name, student-id, and assignment # on the top of the file in the comments.Place all your .cpp files (only) in a folder named your ROLLNUM_SECTION (e.g. 21i-0001_A).
    b. You will zip the folder and submit it on google forms.
    d. No other method of submission will be accepted.
    e. Start the submission process well before time so that you can overcome problems as you face them.

Note: Follow the given instructions to the letter, failing to do so may result in a zero.

Q1. Implementation of Array Class Your goal is to overload the operators for

"Array" class. (40 Marks)

. You will need to write three files (array.h, array.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.

```
class Array {
// think about the private data members...
public:
// provide definitions of following functions...
Array(); // a default constructor
Array(int size); // a parameterized constructor initializing an Array of predefined size

Array(int *arr, int size); // initializes the Array with an existing Array
Array(const Array &); // copy constructor
int& operator[](int i); //returns the integer at index after checking the out of range error
int& operator[](int i)const;
const Array & operator=(const Array&); //copy the array
Array operator+(const Array&); //adds two Array
Array operator-(const Array&); //subtracts two Array
Array operator++(); //adds one to each element of Array
Array operator++(int); //adds one to each element of Array
Array& operator--(int); //subtracts one from each element of array
bool operator==(const Array&)const; //returns true if two arrays are same
bool operator!(); // returns true if the Array is empty
void operator+=(const Array&); //adds two Array
void operator-=(const Array&); //subtracts two Array
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.
~Array(); // destructor...
};
ostream& operator<<(ostream& output, const Array&);//Outputs the Array
istream& operator>>(istream& input, Array&);  //Inputs the Array
```

Q2. Big Integer (60 Marks)

BigInt class is used for the mathematical operations that involve very big integer calculations that are outside the limit of all available primitive data types. For example, a factorial of 100 contains 158 digits in it so we can't store it in any primitive data type available. We can store as large an Integer as we want in it.

Your goal is to overload the operators for a generic "BigInt" class. You will need to write two files (BigInt.h and BigInt.cpp). Your implemented class must fully provide the definitions of following class (interface) functions .

```
class BigInt
{
//think about the private data members
public:
BigInt(int val = 0);
BigInt(const string& text);
BigInt(const BigInt& copy); // copy constructor

// Binary Operators
// Arithmetic Operators
BigInt operator+(const BigInt& val) const;
BigInt operator+(int val) const;
BigInt operator-(const BigInt& val) const;
BigInt operator-(int val) const;
BigInt operator*(const BigInt& val) const;
// Compound Assignment Operators
BigInt operator+=(const BigInt& rhs);
BigInt operator-=(const BigInt& rhs);
BigInt operator*=(const BigInt& rhs);
// Logical Operators
bool operator==(const BigInt& val) const;
bool operator==(const char* val) const;
bool operator!=(const BigInt& val) const;
bool operator<(const BigInt& val) const;
bool operator<=(const BigInt& val) const;
bool operator>(const BigInt& val) const;
bool operator>=(const BigInt& val) const;

// Unary Operators
BigInt& operator++();  // Pre-increment Operator
BigInt operator++(int);  // Post-increment Operator
BigInt& operator--();  // Pre-decrement Operator
BigInt operator--( int ); // Post-decrement Operator
```

```
//Conversion Operator
operator  string();    // return value of the BigInt as string

operator int(); // return the number of digits in big Integer

~BigInt(); // destructor
};


ostream& operator<<(ostream& output, const BigInt& val); // outputs the BigInt
istream& operator>>(istream& input, BigInt& val); // inputs the BigInt
```

Q3. Implementation of a Book (100 Marks)

You will implement 3 classes namely **Line**, **Page** and **Book**. A book will store a certain number of pages and each page will contain a certain number of lines. For example a book can have 10 pages with 20 lines per each page.

All the classes that you will implement must offer the following overloading operators

- Copy Constructors, Default Constructors, Parameterised Constructors and Destructors for each of the class Implementation. (**10 Marks**)
- An overloaded += operator that will add to the existing content of a page. (**40 marks**)
    - Adding a text (char array) to a page would first add the text to the first Available line and the overflowed text should automatically move to the next line.
    - Adding a line (Line object) to a page
    - Adding a page to a book. The existing page must be filled first before moving to the next page.
    - **Note: If adding a word to a line is exceeding the maximum characters (40) in a line the whole word should be part of the next line.**
- An overloaded = that will replace the content of a page/line. (**10 Marks**)
- An overloaded [] to access a specific page/line (**20 Marks**)
    - For Example Book[page][line] -> Book[2][1] should give me the second line on the 3rd page.
- An overloaded + to add 2 specific books. (**15 Marks**)
    - This is not provided in the sample int main() below, however your code should be able to concatenate 2 books and assign it to the 3rd book without editing the content of the first 2 books.
- Overloaded << operator to output the content of the book. (**5 Marks**)

Constraint you must follow

- A line must have a **maximum** of 40 characters.
- A page must have a **maximum** of 20 lines.
- In case of adding content to a page or a line the overflowed text must be written at the next available page/line.

Note: Provide 3 header files (Page.h, Line.h, and Book.h) 4 cpp Files (Page.cpp, Line.cpp, Book.cpp, and BookMain.cpp). Keep the code generic so your program could be tested during demos.

```
int  main()
{// insert code here...
Page p, p2, p3, p4;
p += "I think having someone in your life to look up to is one of the most important things. We all admire people like Linus Torvalds and Bill Gates but trying to make them your role models can be demotivating. Bill Gates began coding at age 13 and formed his first venture at age 17.";
p2 += "Having a local hero or mentor is more helpful. Because you're both living in the same community, there's a greater chance there won't be such a large gap to discourage you. A local mentor probably started coding around the age you did and was unlikely to start a big venture.";
p3 += "First, because their stories seemed like fantasy to me, and second, I couldn't reach them. I chose my mentors and role models to be those near my reach. Choosing a role model doesn't mean you just want to get to where they are and stop. Success is step by step.";
p4 += "You probably can't get one-on-one advice from someone like Bill Gates. You can get the advice they're giving to the public at conferences, which is great, too. I always follow smart people.";
Book b(3);
b += p2;
b += p;
b += p3;
b[2] += p4;
b[2][2] = "I am editing this line using subscripts.";
b[2] += "Adding this text to existing line of page number 3 and overflowed text must go to next line";
Line l("Adding a new line to an existing page.");
Line l2("This is line 2.");
b[1] += l;
b[1] += l2;
b[1] += l;
cout << b;
return 0;
}
```

Output should be the following of the above main:

---------- Page 1----------

Having a local hero or mentor is more
helpful. Because you're both living in
the same community, there's a greater
chance there won't be such a large gap
to discourage you. A local mentor
probably started coding around the age
you did and was unlikely to start a big
Venture. I think having someone in your
life to look up to is one of the most
important things. We all admire people
like Linus Torvalds and Bill Gates but
trying to make them your role models
can be demotivating. Bill Gates began
coding at age 13 and formed his first
venture at age 17. First, because their
stories seemed like fantasy to me, and
second, I couldn't reach them. I chose
my mentors and role models to be those
near my reach. Choosing a role model
doesn't mean you just want to get to

---------- Page 2----------

where they are and stop. Success is
step by step. Adding a new line to an
existing page.This is line 2.Adding a
new line to an existing page.

---------- Page 3----------

You probably can't get one-on-one advice
from someone like Bill Gates. You can
I am editing this line using subscripts.
public at conferences, which is great,
too. I always follow smart people.Adding
this text to existing line of page
number 3 and overflowed text must go to
next line