NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES ISLAMABAD

Object Oriented Programming (CS 1004)

SPRING 2022 ASSIGNMENT # 4

Due Date ≈ Wednesday, May 25th, 2022 (11:59 pm)

Instructions

Submission: Combine all your work in one .zip file. Use proper naming convention for your submission file. Name the .zip file as **SECTION_ROLL-NUM_01.zip** (**e.g. P_21i0412_01.zip**). Your zip file should not contain any folders or subfolders. It should only contain .cpp or .h files for each question, if additional files are asked they will be mentioned with each question. Submit .zip file on Google Classroom within the deadline. Failure to submit according to the above format would result in **25% marks deduction**. Submissions on the email will not be accepted.

Plagiarism: Plagiarism cases will be dealt with strictly. If found plagiarized, both the involved parties will be awarded zero marks in this assignment, all the remaining assignments, or even an **F grade** in the course. Copying from the internet is the easiest way to get caught!

Deadline: The deadline to submit the assignment is 25th **May 2022 at 11:59 PM**. Late submission with marks deduction will be accepted according to the course policy shared earlier. Correct and timely submission of the assignment is the responsibility of every student; hence no relaxation will be given to anyone.

Bonus: In case you implement any additional feature which you think is worth of bonus, make it prominent so that we can see it at runtime.

Note:

- *Each question will be graded on the basis of your effort, additional marks will be awarded for using good programming practices, including: memory efficient programs, well-written, good design and properly commented.*
- All programs must be generic.
- You can change the argument, return type and also add new data members in the given structures.
- Follow the given instructions to the letter, failing to do so will result in a zero.

Q 1: Implementation of Extended Character – Your goal is to implement "ExtendedCharacter" class. Primitive data type to store character needs 1 Byte ranging from (0000 0000 TO 1111 1111). Your goal it to implement character class that will able to store characters in their binary form as strings (0000000000000000 TO 1111111111111111). Each instance will be stored in binary representation e.g. 'A' will be stored as "0000000001000001". You will need to write three files (ExtendedCharacter.h, ExtendedCharacter.cpp and Q1.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 ExtendedCharacter{
  // think about the private data members...
  public:
    //include all the necessary checks before performing the operations in
    the functions
    ExtendedCharacter();// a default constructor
    ExtendedCharacter (int);// a parametrized constructor
    ExtendedCharacter (char); // a parametrized constructor
    void set(char);//set value

    toString(char);//write your own function that converts a character into
    its binary equivalent and return it as string, you can use this function
    in the parameterized constructor when constructing the object

    ExtendedCharacter& operator=(const ExtendedCharacter &); //
    ExtendedCharacter& operator+(const ExtendedCharacter &); //
    ExtendedCharacter& operator-(const ExtendedCharacter &); //

    bool operator==(const ExtendedCharacter &); //
    bool operator!=(const ExtendedCharacter &); //
    bool operator<=(const ExtendedCharacter &); //
    bool operator>=(const ExtendedCharacter &); //
    bool operator<(const ExtendedCharacter &); //
    bool operator>(const ExtendedCharacter &); //
    ExtendedCharacter& operator++(); //
    ExtendedCharacter& operator++(int); //
    ExtendedCharacter& operator--(); //
    ExtendedCharacter& operator--(int); //

    operator int(); //converts the ExtendedCharacter into an integer
    operator short(); //converts the ExtendedCharacter into an integer

    void operator()(int, int); //Given a range in integers determine if the
    character is an alphabet (lower or upper), number

};
```

```
int main(){
}
```

128	Ç	144	É	160	á	176	▒	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	161	í	177	▒	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	162	ó	178	▓	195	├	211	╙	227	π	243	≤
131	â	147	ô	163	ú	179	│	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	164	ñ	180	┤	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	165	Ñ	181	╡	198	╞	214	╓	230	µ	246	÷
134	å	150	û	166	ª	182	╢	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	167	º	183	╖	200	╚	216	╪	232	Φ	248	°
136	ê	152	_	168	¿	184	╕	201	╔	217	┘	233	Θ	249	·
137	ë	153	Ö	169	_	185	╣	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	170	¬	186	║	203	╦	219	▉	235	δ	251	√
139	ï	156	£	171	½	187	╗	204	╠	220	▄	236	∞	252	_
140	î	157	¥	172	¼	188	╝	205	═	221	▌	237	ϕ	253	²
141	ì	158	_	173	¡	189	╜	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	174	«	190	╛	207	╧	223	▀	239	∩	255	
143	Å	192	└	175	»	191	┐	208	╨	224	α	240	≡		

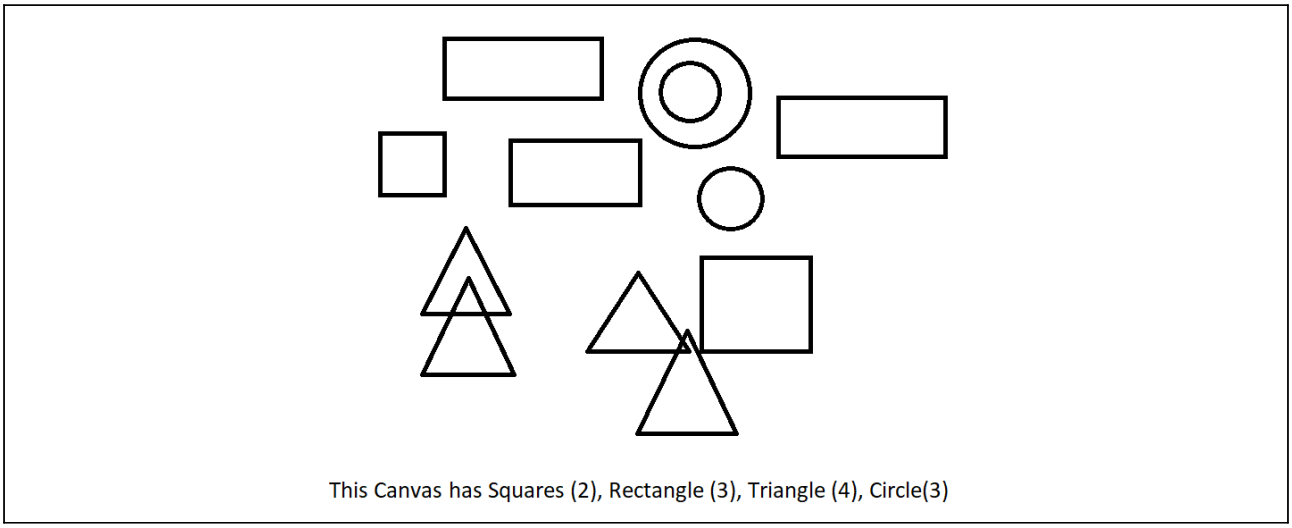
Q 2: Implementation of Matrix Class – Your goal is to implement a generic “Matrix" class. You will need to write three files (matrix.h, matrix.cpp and Q2.cpp). Your implemented class must fully provide the definitions of following class (interface) functions. Please also write down the test code to drive 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 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

    operator[]();//set and get value at (i,j)

    Matrix& operator=(const Matrix &); //assigns matrix on RHS to the one on
    LHS
    Matrix& operator+(const Matrix &); //add two matrices
    Matrix& operator-(const Matrix &); //subtracts two matrices
    Matrix& operator*(const Matrix &); //dot product of two matrices

    bool operator==(const Matrix &); //checks if two matrices are equal
```

```
    Matrix& operator+=(int);//adds an integer to all elements
    Matrix& operator-=(int);//subtracts an integer from all elements
    Matrix& operator*=(int);//multiples an integer to all elements
    Matrix& operator/=(int);//divides all elements by an integer

    Matrix& operator()(int A, int B);//returns a new matrix of size A x B

    ~Matrix();
};

ostream& operator<<(ostream&, const Matrix &); //outputs the matrix
istream& operator>>(istream&, const Matrix &); //inputs the matrix

int main(){
    //write test code
    Matrix M1(3,3), M2(3,3), M3(3,3); //declare multiple 3X3 matrices

    Matrix M4 = M1 – M2 + M3; //should be able to write expressions of this
    form

    Matrix M5 = M1(1,2); //returns a matrix of size 1X2 starting from row-0
    and col-0
}
```

Q 5: Canvas Class – You will have to implement a Canvas class which may contain multiple shapes. In this question you have to use the Shape class that you have already implemented in the previous assignment. You will separate your code in three files (Canvas.h, Canvas.cpp, and Q5.cpp). A canvas can have multiple shapes but since we have implemented Shape class earlier we would like to store information of different types of shapes separately. This information will be stored in the Canvas class. A sample canvas is given below, an instance of your canvas should be able to store this information.

[Hint]: We want to store different shapes separately



This Canvas has Squares (2), Rectangle (3), Triangle (4), Circle(3)

Figure: Sample Canvas

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 Shape{
   //use the student class that you have already implemented in the previous
   assignment


   operator[]();//whenever you have to access any information of shape
   operator+=(Shape&);
};

const int TYPE_OF_SHAPES_PER_CANVAS = 10;

class Canvas{
  private:
   // think of private data member carefully
  public:
   //implement necessary constructors
   operator[](Shape&); //returns the appropriate reference to add or remove
   the
   shape from the canvas
   operator[](string); //Returns the shapes of a certain type “Circle” – all
   circles
   operator()(int, int); //prints the information of specific shape in the
   canvas

   operator+(Canvas&); //adds two canvas and creates a third one with shapes
   of first two canvases – be careful of the limit of shapes per canvas

   operator string(); //converts the canvas information into a well-
   formatted string. It should include information of all types and shapes
   in it.
   ~Canvas();
};
int main(){
   Canvas C;
   Shape s_Obj(2, 2, 10); //circle made at center 2,2 and radius=10
   C["Circle"] += s_Obj; //should add the object to the canvas
   C(0,1); //prints a certain shape in the canvas
   Canvas C1;
   C1[“Rectangle”] = Shape(3, 4, 5, 10); //rectangle is added to Canvas C1
   Canvas C3;
   C3 = C1 + C2; //Canvas C3 will now have a circle and a rectangle
}
```