Object Oriented Programming (OOP)
Assignment 1

1. Create a class named `Product`

- It should have the following three **private** attributes: `productName` (**string**), `price` (**float**), `quantity` (**int**).
- In the `main()` function, create a 2D Array of `Product` objects representing a 2×2 grid in a shop (2 shelves with 2 products each). Initialize all `Product` object with values.
- Then, write a method to display the details of each product in the 2D Array. Make sure the display output is formatted neatly.

Ans. Here is the C++ program of a class named `Product` that satisfies the conditions above:

```cpp
/**
 * ======================
 * Name: Shadman Shahriar
 * ID  : 20245013408
 * ======================
 */

#include <iostream>
using namespace std;

class Product
{
    string productName;
    float price;
    int quantity;

public:
    Product(string n, float p, int q)
    {
        productName = n;
        price = p;
        quantity = q;
```

```
    }

    void display(int i)
    {
        cout << i << endl;
        cout << "Product : " << productName << endl;
        cout << "Price   : " << price << "/=" << endl;
        cout << "Quantity: " << quantity << endl;
        cout << endl;
    }
};

int main()
{
    const int row = 2;
    const int column = 2;
    Product shop[row][column] = {
        {Product("Toast Biscuit", 110, 2), Product("Soap", 50,
        4)},
        {Product("Shampoo", 180, 1), Product("Kitkat", 105,
        1)}};

    for (int r = 0; r < row; r++)
        for (int c = 0; c < column; c++)
            shop[r][c].display((c + 1) + (row * r));

    return 0;
}
```

2. Create a class named `Book`

- It should have **private** attributes: `title` (**string**), `author` (**string**), `pages` (**int**).
- Write a function `assignBook()` that assigns values to a `Book` object. Also, write another function `displayBook()` to display its details.
- Next, create a function named `compareBooks()` , **outside the class**, that takes two `Book` object as pointers and returns the one with more pages.
    - Use `assignBook()` to initialize two `Book` object.

- Use `compareBooks()` to find and **display the book** with the higher page count.

Hint: Focus on using *object pointers* when passing the objects to the functions (Here, `pages` is a **private** member, then what do you need to access it from outside the class?)

Ans. Here is the C++ program of a class named `Book` that satisfies the conditions above:

```cpp
/**
 * ======================
 * Name: Shadman Shahriar
 * ID  : 20245013408
 * ======================
 */

#include <iostream>
using namespace std;

class Book
{
    string title;
    string author;
    int pages;

public:
    void assignBook(string t, string a, int p)
    {
        title = t;
        author = a;
        pages = p;
    }

    void displayBook()
    {
        cout << "Name   :  " << title << endl;
        cout << "Author:  " << author << endl;
        cout << "Pages  :  " << pages << endl;
    }
```

```
    int getPages()
    {
        return pages;
    }
};

Book *compareBooks(Book *b1, Book *b2)
{
    if (b1->getPages() > b2->getPages())
        return b1;
    else
        return b2;
}

int main()
{
    Book book1, book2;
    book1.assignBook("Atomic Habits", "James Clear", 320);
    book2.assignBook("Show Your Work!", "Austin Kleon", 224);

    cout << "Book 1:" << endl;
    book1.displayBook();

    cout << endl;
    cout << "Book 2:" << endl;
    book2.displayBook();

    Book *bookMaxPage = compareBooks(&book1, &book2);

    cout << endl;
    cout << "Book with more pages:" << endl;
    bookMaxPage->displayBook();
    return 0;
}
```

3. Design two classes `Rectangle` and `Square`

- Both classes should have **private** data members for their dimensions (**length** and **width** for `Rectangle`, **side** for `Square`)

- Create a **friend function** named `printArea()` that takes both a `Rectangle` object and a `Square` object and prints their areas.
- Create another **friend function** named `increaseSide()` that is a member function of `Square` and a friend of `Rectangle`. This function should increase the **side** length of the `Square` object by the value of the `Rectangle`'s **length**.
- In the `main()` function, create objects for both `Rectangle` and `Square`, initialize them, and demonstrate the use of both **friend functions**.

Ans. Here is a C++ program with two classes named `Rectangle` and `Square` that follows the conditions above:

```cpp
/**
 * ======================
 * Name: Shadman Shahriar
 * ID  : 20245013408
 * ======================
 */

#include <iostream>
using namespace std;

class Rectangle;
class Square
{
    int side;

public:
    Square(int s) : side(s) {}
    friend void printArea(Rectangle rect, Square sq);
    void increaseSide(Rectangle &rect);
};

class Rectangle
{
    int length;
    int width;

public:
    Rectangle(int l, int w) : length(l), width(w) {}
```

```
    friend void printArea(Rectangle rect, Square sq);
    friend void Square::increaseSide(Rectangle &rect);
};

void Square::increaseSide(Rectangle &rect)
{
    side += rect.length;
}

void printArea(Rectangle rect, Square sq)
{
    int areaRect = rect.length * rect.width;
    int areaSq = sq.side * sq.side;

    cout << "Rectangle: " << areaRect << " sq units" << endl;
    cout << "Square:    " << areaSq << " sq units" << endl;
}

int main()
{
    Rectangle rect(10, 20);
    Square sq(15);

    cout << "Area before increasing side length:" << endl;
    printArea(rect, sq);

    cout << endl;
    cout << "Area after increasing side length:" << endl;
    sq.increaseSide(rect);
    printArea(rect, sq);
}
```

4. Write the basic differences and similarities between Class, Structure and Union.

Ans. Here are the differences and similarities between **Class**, **Structure**, and **Union** in C++:

	Class	Structure	Union
Keyword	It is declared using the **class** keyword.	It is declared using the **struct** keyword.	It is declared using the **union** keyword.
Use Case	It is normally used for data abstraction and inheritance.	It is normally used for the grouping of different datatypes.	It is used to achieve memory efficiency when the available memory is limited.
Accessing Members	Members of a class are **private** by default.	Members of a structure are **public** by default.	Member variables of a union are **public** by default.
Size	Theoritically, the size of a class can be up to 2 to the power of 55. (2^55)	The size of a structure is the sum of the size of its data members.	The size of a union is equal to the size of the largest data type.
Memory Allocation	It can store multiple values of the various members.	Same as Class.	It stores one value at a time for all of its members.
Value Altering	Altering the values of a single member does not affect the other	Altering the values of a single member does not affect the other	Altering the values of a single member, it affects the

	Class	Structure	Union
	members of a Class.	members of a Structure.	values of other members.
Initialization	We can initialize multiple members at the same time.	Same as Class.	We can only initiate the first member at a time.

References

- **GeeksforGeeks**: Difference Between Structure and Class in C++
- **Byjus**: Difference Between Structure and Union in C
- **Itanium C++ ABI draft**: Section 1.2

Code

You can find all the code snippets **here**.