COMP710: Studio Session 02 – Exercise:

EXERCISE: *Using Enumerations*

Add a new C++ Project named "Using Enumerations" to your "SS02" Visual Studio Solution for this exercise. Create a **`main.cpp`** and add the following source code to the file:

```
#include <iostream>

enum PLAYER_RANK
{
    PR_IRON,
    PR_BRONZE,
    PR_SILVER,
    PR_GOLD,
    PR_PLATINUM,
    PR_DIAMOND
};

enum MEMBERSHIP_TIER
{
    MT_GOLD,
    MT_SILVER,
    MT_FREE,
    MT_BANNED
};

void DisplaySizeChecks();

void PrintMembership(MEMBERSHIP_TIER membershipTier);
MEMBERSHIP_TIER UpgradeMembershipOneTier(MEMBERSHIP_TIER current);

void PrintPlayerRank(PLAYER_RANK playerRank);
void PromotePlayer(PLAYER_RANK& playerRank);
void DemotePlayer(PLAYER_RANK& playerRank);

int main()
{
    MEMBERSHIP_TIER exampleMembership = MT_FREE;

    PrintMembership(exampleMembership);

    exampleMembership = UpgradeMembershipOneTier(exampleMembership);

    PrintMembership(exampleMembership);

    DisplaySizeChecks();

    return 0;
}

void DisplaySizeChecks()
{
    std::cout << "sizeof(PLAYER_RANK) is " << sizeof(PLAYER_RANK);
    std::cout << " bytes." << std::endl;
    std::cout << "sizeof(MEMBERSHIP_STATUS) is ";
    std::cout << sizeof(MEMBERSHIP_TIER) << " bytes." << std::endl;
}
```

```
void DisplaySizesChecks()
{
    std::cout << "sizeof(PLAYER_RANK) is " << sizeof(PLAYER_RANK);
    std::cout << " bytes." << std::endl;
    std::cout << "sizeof(MEMBERSHIP_STATUS) is ";
    std::cout << sizeof(MEMBERSHIP_TIER) << " bytes." << std::endl;
}

MEMBERSHIP_TIER UpgradeMembershipOneTier(MEMBERSHIP_TIER current)
{
    if (current > MT_GOLD)
    {
        current = static_cast<MEMBERSHIP_TIER>(current - 1);
    }

    return current;
}

void PrintMembership(MEMBERSHIP_TIER membershipTier)
{
    switch (membershipTier)
    {
    case MT_GOLD:
        std::cout << "Member is gold level!" << std::endl;
        break;
    case MT_SILVER:
        std::cout << "Member is silver level." << std::endl;
        break;
    case MT_FREE:
        std::cout << "Member is freemium." << std::endl;
        break;
    case MT_BANNED:
        std::cout << "Member is banned!" << std::endl;
        break;
    default:
        std::cout << "Unknown membership!" << std::endl;
        break;
    }
}
```

Compile, link and run the above program source code, the output should be exactly as follows:

```
Member is freemium.
Member is silver level.
sizeof(PLAYER_RANK) is 4 bytes.
sizeof(MEMBERSHIP_STATUS) is 4 bytes.
```

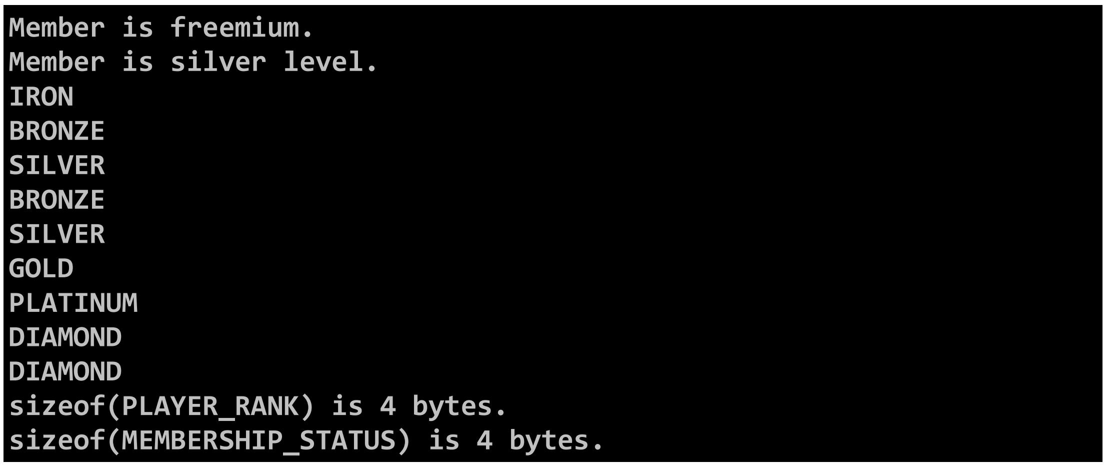
Note the size of each enumerated type in this program is 4 bytes. Also note that the enumerated constant values for **MEMBERSHIP_TIER** are each prefixed with the **MT_** naming style, and the **PLAYER_RANK** is similarly prefixed with **PR_**. This is a usual programming practice in C++ prior to C++11, without this practice, this program would attempt to declare two enumerated constant values for the identifier **GOLD**, which would cause a compiler error due to the name conflict.

Firstly, implement the functions **PrintPlayerRank**, **PromotePlayer**, and **DemotePlayer**. The prototype declarations for each of these has been made above the **main** function, so do not change these signatures (the return type, function name, or parameter list). Implement the function definitions for these three functions after the **PrintMembership** function definition. Review the implementations of **PrintMembership** and **UpgradeMembershipOneTier** for inspiration, however, note that the **PromotePlayer** and **DemotePlayer** functions take in a **PLAYER_RANK** by reference – so these two functions will use pass-by-reference behaviour, altering the caller's data via the parameter, rather than simply returning a value via return type, as in the case of the **UpgradeMembershipOneTier** function.

Next, add some test code into the main function prior to the **DisplaySizeChecks** call to experiment with creating an enumerated type variable of **PLAYER_RANK** type, and then pass this into the **PrintPlayerRank**, **PromotePlayer**, and **DemotePlayer**, functions to robustly test their functionality.

Build your project and check the output makes sense – and again check that your testing of your new player rank printing, player promotion and demotion functions work as intended. Test your program robustly to ensure the output matches your testing expectations.

An example of the completed program may be as follows, however your test cases may be different:



```
Member is freemium.
Member is silver level.
IRON
BRONZE
SILVER
BRONZE
SILVER
GOLD
PLATINUM
DIAMOND
DIAMOND
sizeof(PLAYER_RANK) is 4 bytes.
sizeof(MEMBERSHIP_STATUS) is 4 bytes.
```