CSE Assignment (C++)

Problem 1

Question:

> Overloading +, == operator to add and compare two distance objects

Code:

```
#include <iostream>
#include <math.h>

using namespace std;

class Dist
{
    private:

        double distance;

    public:

        Dist()
        {
            distance = 0;
        }

        Dist(const double &dist)
        {
            distance = dist;
        }

        double gd() const
        {
            return distance;
        }

        Dist operator +(const Dist &dist) const
        {
```

```
            return Dist(distance + dist.gd());
        }

        bool operator ==(const Dist &dist) const
        {
            return (distance == dist.gd()) ? true : false;
        }
};

int main()
{
    Dist A(7), B(7);

    cout << (A + B).gd() << endl;
    cout << (A == B) << endl;

    return 0;
}
```

Output:

```
14
1
```

Problem 2

Question:

> Overloading == operators to compare two strings

Code:

```
#include <iostream>
#include <string.h>

using namespace std;

class SCmp
{
    private:
```

```
        char cmpstring[20];

    public:

        SCmp(char string[])
        {
            strcpy(cmpstring, string);
        }

        char* gs()
        {
            return cmpstring;
        }

        bool operator ==(SCmp &S)
        {
            return strcmp(cmpstring, S.gs()) == 0 ? true : false;
        }
};

int main()
{
    char hm[] = "Apple";
    SCmp A(hm), B(hm);

    cout << (A == B) << endl;

    return 0;
}
```

Output:

```
1
```

Problem 3

Question:

> Overload ++(prefix), +=, *= for matrix objects

Code:

```
#include <iostream>

using namespace std;

class Matrix
{
    private:

        int grid[3][3];

    public:

        Matrix()
        {
            for (int i = 0; i < 3; i++)
            {
                for (int j = 0; j < 3; j++)
                {
                    grid[i][j] = 0;
                }
            }
        }

        Matrix(int (&cp)[3][3])
        {
            for (int i = 0; i < 3; i++)
            {
                for (int j = 0; j < 3; j++)
                {
                    grid[i][j] = cp[i][j];
                }
            }
        }

        void dispMatrix()
        {
            for (int i = 0; i < 3; i++)
            {
                cout << endl;

                for (int j = 0; j < 3; j++)
                {
                    cout << grid[i][j] << " ";
                }
            }
        }
```

```
    int gelem(int i, int j)
    {
        return grid[i][j];
    }

    Matrix operator ++()
    {
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                grid[i][j]++;
            }
        }

        return *this;
    }

    void operator +=(Matrix &op2)
    {
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                grid[i][j] += op2.gelem(i, j);
            }
        }
    }

    void operator *=(Matrix &op2)
    {
        int prod[3][3];

        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                prod[i][j] = 0;

                for (int k = 0; k < 3; k++)
                {
                    prod[i][j] += grid[i][k] * op2.gelem(k, j);
                }
            }
        }

        for (int i = 0; i < 3; i++)
        {
```

```
                for (int j = 0; j < 3; j++)
                {
                    grid[i][j] = prod[i][j];
                }
            }
        }
};

int main()
{
    int arr[3][3] = {{1, 2, 3}, {1, 2, 3}, {1, 2, 3}};

    Matrix m1(arr), m2(arr);

    (++m1).dispMatrix();

    cout << endl << endl;

    m1 += m2;

    m1.dispMatrix();

    cout << endl << endl;

    m1 *= m2;

    m1.dispMatrix();

    cout << endl;

    return 0;
}
```

Output:

```
2 3 4
2 3 4
2 3 4


3 5 7
3 5 7
3 5 7


15 30 45
```

```
15 30 45
15 30 45
```

Problem 4

Question:

> Program to show how a compiler explicitly converts basic data types

Code:

```cpp
#include <iostream>

using namespace std;

class Complex
{
    private:

        double real;
        double imag;

    public:

        explicit Complex(double r = 0.0, double i = 0.0):real(r), imag(i)
        {}

        bool operator == (Complex rhs)
        {
            return (real == rhs.real && imag == rhs.imag);
        }
};

int main()
{
    Complex com1(3.0, 0.0);

    if (com1 == (Complex)3.0)
    {
```

```
        cout << "Same";
    }
    else
    {
        cout << "Not Same";
    }

    return 0;
}
```

Output:

```
Same
```

Problem 5

Question:

> Program to demonstrate the conversion from user defined data type to basic data type

Code:

```
#include <iostream>
#include <math.h>

using namespace std;

class Complex
{
    private:

        double real;
        double imag;

    public:

        Complex(double r = 0.0, double i = 0.0):real(r), imag(i){}

        operator double()
```

```
        {
            return sqrt(pow(real, 2) + pow(imag, 2));
        }
};

int main()
{
    Complex com1(3.0, 3.0);
    double x;

    x = com1;

    cout << x;

    return 0;
}
```

Output:

```
4.24264
```

Problem 6

Question:

> Program to convert between two user defined classes – class of type feet to class of type inches and vice versa

Code:

```
#include <iostream>

using namespace std;

class TC2;
class TC1
{
private:
    int t;
```

```
public:
    TC1()
    {
        t = 5;
    }

    int gt()
    {
        return t;
    }

    operator TC2();
};

class TC2
{
private:
    int tt;

public:
    TC2()
    {
        tt = 6;
    }

    int gt()
    {
        return tt;
    }

    operator TC1()
    {
        TC1 y;
        return y;
    }
};

TC1::operator TC2()
{
    TC2 x;
    return x;
}

int main(int argc, char const *argv[])
{
    TC1 a;
    TC2 b;
```

```
    cout << a.gt();
    cout << b.gt();

    TC1 c;

    cout << ((TC2)c).gt();

    TC2 d;

    cout << ((TC1)d).gt();

    return 0;
}
```

Output:

```
5665
```
