```
//*****************************************************************************************
**
//
//              File:                                   student.h
//
//              Student:                                Sean Herrick
//
//              Assignment:                             Program #08
//
//              Course Name:                    Data Structures II
//
//              Course Number:                  COSC 3100-01
//
//              Due:                                    October 7th, 2022
//
//
//              This program displays, prints, and enters in student data.
//
//              Other files required:
//                      1.      studentList.h
//                      2.      HashTable.h
//                      3.      Node.h
//
//*****************************************************************************************
**

#ifndef STUDENT_H
#define STUDENT_H

//*****************************************************************************************
**

#include "Node.h"

//*****************************************************************************************
**

struct Student
{
        int id,
                year,
                credits;
        char name [ 50 ],
                citystate [ 50 ],
                phone [ 12 ],
                gender,
                major [ 6 ];
        float gpa;

        Student ( );
        friend ostream & operator << ( ostream & out, const Student& data );
        bool operator == ( const Student & rhs ) const;
        bool operator == ( int value ) const;
        bool operator != ( const Student & rhs ) const;
        bool operator != ( int value ) const;
        bool operator < ( const Student & rhs ) const;
        bool operator < ( int value ) const;
        bool operator > ( const Student & rhs ) const;
        bool operator > ( int value ) const;
        bool operator <= ( const Student & rhs ) const;
        bool operator <= ( int value ) const;
        bool operator >= ( const Student & rhs ) const;
        bool operator >= ( int value ) const;
        int operator % ( int value ) const;
```

```
        Student & operator = ( int value );

};

//*****************************************************************************************
**

Student :: Student ( )
{
        id = 0;
}

//*****************************************************************************************
**

ostream & operator << ( ostream & out, const Student & data )
{
        out << data.id << "/";

        for ( int i = 0; i < 6; i++ )
        {
                out << data.name [ i ];
        }

        return out;
}

//*****************************************************************************************
**

bool Student :: operator == ( const Student & rhs ) const
{
        return ( this->id == rhs.id );
}

//*****************************************************************************************
**

bool Student :: operator == ( int value ) const
{
        return ( this->id == value );
}

//*****************************************************************************************
**

bool Student :: operator != ( const Student & rhs ) const
{
        return ( this->id != rhs.id );
}

//*****************************************************************************************
**

bool Student :: operator != ( int value ) const
{
        return ( this->id != value );
}

//*****************************************************************************************
**

bool Student :: operator < ( const Student & rhs ) const
{
        return ( this->id < rhs.id );
```

```
}

//*****************************************************************************************
**

bool Student :: operator < ( int value ) const
{
        return ( this->id < value );
}

//*****************************************************************************************
**

bool Student :: operator > ( const Student & rhs ) const
{
        return ( this->id > rhs.id );
}

//*****************************************************************************************
**

bool Student :: operator > ( int value ) const
{
        return ( this->id > value );
}

//*****************************************************************************************
**

bool Student :: operator <= ( const Student & rhs ) const
{
        return ( this->id <= rhs.id );
}

//*****************************************************************************************
**

bool Student :: operator <= ( int value ) const
{
        return ( this->id <= value );
}

//*****************************************************************************************
**

bool Student :: operator >= ( const Student & rhs ) const
{
        return ( this->id >= rhs.id );
}

//*****************************************************************************************
**

bool Student :: operator >= ( int value ) const
{
        return ( this->id >= value );
}

//*****************************************************************************************
**

Student & Student :: operator = ( int value )
{
        this->id = value;
```

```
        return *this;
}

//*****************************************************************************************
**

int Student :: operator % ( int value ) const
{
        return ( this->id % value );
}

//*****************************************************************************************
**

#endif

//*****************************************************************************************
**

/*
*/
```