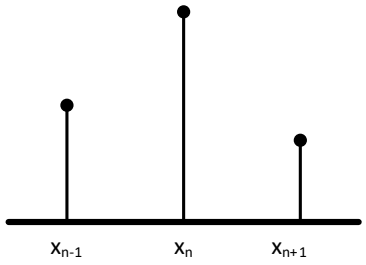
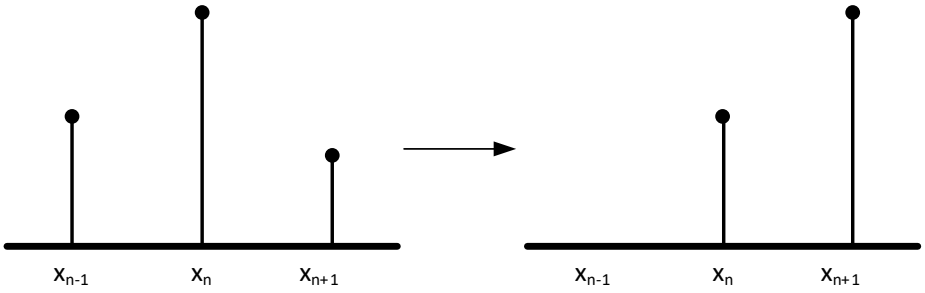
1) Assume that we have a discrete signal as follows:



Write a class Discrete Signal Algebra (DSA) with the following capabilities:

a) Define three **private double** data member (i.e xNM1, xN, and xNP1).
b) The constructor of the class is default constructor with values 0.0 for xNM1, xN, and xNP1.
c) Include overloaded the addition, subtraction, multiplication and division operators for this class. **Overloaded operators must be public member functions.** These operators perform pair-wise operations. For example, we will add xNM1, xN, and xNP1 data members of two objects for addition. **For division, implement appropriate exception handling statements to handle zero division.**
d) Include preincrement and postincrement operators for this class. **Overloaded operators must be public member functions.** For these operators, shift the values 1 item to the right as shown in the figure:



e) Include overloaded the equality and inequality operators (==, !=) for this class. **Overloaded operators must be non-member functions.** If the all data members of two objects are equal, in that case these two objects are equal.
f) Include overloaded the less than and greater than operators (<, >) for this class. **Overloaded operators must be non-member functions.** For these operators, calculate following formula:

$$mag = \sqrt[2]{x_{NM1}^2 + x_N^2 + x_{NP1}^2}$$

If mag is greater than for an object, the greater than operator will return true. Similarly, if mag is less than for an object, the less than operator will return true.

g) Include overloaded the stream extraction operator function operator (>>) for this class. **The format for input must be xNM1---xN---xNP1. Assume that inputs for xNM1, xN, and xNP1could be positive and one-digit.**
h) Include overloaded the stream insertion operator function operator (<<) for this class. **The format for output must be xNM1…xN…xNP1.**

Warning: You must use const keyword and reference parameters in appropriate situations.

Test your program with following driver program.

```
main()
{
    DSA d1,d2,d3,x;

    cout << "Enter discrete signals in the form xNM1---xN---xNP1" << endl;
    cin >> d1 >> d2;

    cout << d1 << endl << d2 << endl << x << endl;

    x=d1+d2;
    cout << d1 << " + " << d2 << " = " << x << endl;

    x=d1-d2;
    cout << d1 << " - " << d2 << " = " << x << endl;

    x=d1*d2;
    cout << d1 << " * " << d2 << " = " << x << endl;

    try
    {
        x=d1/d2;
        cout << d1 << " / " << d2 << " = " << x << endl;
    }
    catch(invalid_argument &e)
    {
        cout << "\nException: " << e.what() << endl << endl;
    }

    try
    {
        x=d1/d3;
        cout << d1 << " / " << d2 << " = " << x << endl;
    }
    catch(invalid_argument &e)
    {
        cout << "\nException: " << e.what() << endl << endl;
    }

    cout << d1++ << endl << ++d2 << endl;

    if (d1==d2)
        cout << "d1 is equal to d2" << endl;

    if (d1!=d2)
        cout << "d1 is not equal to d2" << endl;

    if (d1>d2)
        cout << "d1 is greater than d2" << endl;

    if (d1<d2)
        cout << "d1 is less than d2" << endl;
}
```

```
Enter discrete signals in the form xNM1---xN---xNP1
3---9---1
6---3---4
3...9...1
6...3...4
0...0...0
3...9...1 + 6...3...4 = 9...12...5
3...9...1 - 6...3...4 = -3...6...-3
3...9...1 * 6...3...4 = 18...27...4
3...9...1 / 6...3...4 = 0.5...3...0.25

Exception: Zero Division

3...9...1
0...6...3
d1 is not equal to d2
d1 is greater than d2
```

2) Write a class Sine with the following capabilities:

a) Define two **private double** data member (i.e angle and value).
b) The constructor of the class is default constructor with values 0.0 for angle and value.
c) Include overloaded the addition, subtraction, and multiplication operators for this class. **Overloaded operators must be public member functions.** For these operators, first, calculate the angle data member according to the operation. Then, calculate value data member using given formulas. For addition use following formula:

$$\sin(A+B) = \sin A \cos B + \cos A \sin B$$

For subtraction use following formula:

$$\sin(A-B) = \sin A \cos B - \cos A \sin B$$

For multiplication use following formula:

$$\sin A \sin B = \frac{\sin\big(90-(A-B)\big) - \sin\big(90-(A+B)\big)}{2}$$

d) Include preincrement and postincrement operators for this class. **Overloaded operators must be public member functions.** For these operators, add 1 degree to angle data member and also update the value data member.

e) Include overloaded the equality and inequality operators (==, !=) for this class. **Overloaded operators must be non-member functions.** If the angle data member of two objects is equal, in that case these two objects are equal.

f) Include overloaded the less than and greater than operators (<, >) for this class. **Overloaded operators must be non-member functions.** If value data member is greater than for an object, the greater than operator will return true. Similarly, if value data member is less than for an object, the less than operator will return true.

g) Include overloaded the stream extraction operator function operator (>>) for this class. **The format for input must be sin angle. Assume that input for angle must be in terms of degree. Also, the angle must be between 0-90 degrees. For valid input, implement appropriate exception handling statements to handle invalid argument case. Notice that, we do not enter value data member. Therefore, in this function, if the angle is valid, then calculate the sine value and assign the value data member.**

h) Include overloaded the stream insertion operator function operator (<<) for this class. **The format for output must be sin(angle)= value.**

Warning: You must use const keyword and reference parameters in appropriate situations.

Test your program with following driver program.

```
main()
{
    Sine s1,s2,s3,x;

    try
    {
        cout << "Enter angle in terms of degree in the form sin angle" << endl;
        cin >> s1 >> s2;
    }
    catch(invalid_argument &e)
    {
        cout << "\nException: " << e.what() << endl << endl;
    }

    cout << s1 << endl << s2 << endl << x << endl;

    x=s1+s2;
    cout <<  "sin(s1 + s2)=====>" << x << endl;

    x=s1-s2;
    cout <<  "sin(s1 - s2)=====>" << x << endl;

    x=s1*s2;
    cout <<  "sin(s1)sin(s2)=====>" << x << endl;

    cout << s1++ << endl << ++s2 << endl;


    if (s1==s2)
        cout << "s1 is equal to s2" << endl;

    if (s1!=s2)
        cout << "s1 is not equal to s2" << endl;

    if (s1>s2)
        cout << "s1 is greater than s2" << endl;

    if (s1<s2)
        cout << "s1 is less than s2" << endl;

    try
    {
        cout << "Enter angle in terms of degree in the form sin angle" << endl;
        cin >> s3;
    }
    catch(invalid_argument &e)
    {
        cout << "\nException: " << e.what() << endl << endl;
    }
}
```

```
Enter angle in terms of degree in the form sin angle
sin 45
sin 30
sin(45)= 0.707107
sin(30)= 0.5
sin(0)= 0
sin(s1 + s2)=====>sin(75)= 0.965926
sin(s1 - s2)=====>sin(15)= 0.258819
sin(s1)sin(s2)=====>sin(20.7048)= 0.353553
sin(45)= 0.707107
sin(31)= 0.515038
s1 is not equal to s2
s1 is greater than s2
Enter angle in terms of degree in the form sin angle
sin 100

Exception: Invalid argument
```