Flower Shop Order

1. Create a new class that represents a *Flower Shop Order*.
2. Add a *bouquet type*, *number of roses*, and *subtotal* property to the Flower Shop Order class:
    - `bouquet type`: indicates the type of bouquet.
    - `number of roses`: indicates the number of roses added to the bouquet.
    - `subtotal`: indicates the order subtotal before shipping by adding $19.99 for the standard bouquet, plus $2.99 for each rose.
3. Create a constructor that accepts `bouquet type` and `number of roses`.
4. Instantiate an object, or objects, in `Main()`, and use the object(s) to test your methods.
5. Create a method that calculates the delivery total using input parameters of a `bool` for `sameDayDelivery` and a string `zipCode`:
    - The delivery fee is $3.99 if the zip code falls between 20000 and 29999 (+$5.99 for same-day delivery).
    - The delivery fee is $6.99 if the zip code falls between 30000 and 39999 (+$5.99 for same-day delivery).
    - The delivery fee is waived ($0.00) if the zip code falls between 10000 and 19999.
    - All other zip codes cost $19.99 (same-day delivery is not an option).
6. Override the `ToString()` method and have it return `"ORDER - {bouquet type} - {number of roses} roses - {subtotal}"` where `{bouquet type}`, `{number of roses}`, and `{subtotal}` are placeholders for the actual values. The values from the object should be shown in the string where `{variable-name}` is indicated.
7. Implement unit tests to validate the functionality of:
    - The correct subtotal calculation
    - The delivery fee calculation
8. In the Program class, within the Main method, read in the provided csv file `FlowerInput.csv`, and use it to populate a list of *Flower Shop Order* objects.
9. Add up the subtotal total for all of the orders in the list, and print it to the screen.