```
#include <iostream>
#include <string>
using namespace std;

class Television {

    /* class-level attributes */
    static const int MIN_VOLUME = 0;
    static const int MAX_VOLUME = 10;
    static const int MIN_CHANNEL = 2;
    static const int MAX_CHANNEL = 99;

private:
    // Data members of instance

    /** Whether the power is on */
    bool    powerOn;

    /** Whether the tv is muted */
    bool    muted;

    /** The current volume level */
    int     volume;

    /** The most recent previous channel number */
    int     prevChan;

public:

    /** Creates a new Television instance.
     *
     *  The power is initially off.  Upon the first time the TV is turned on,
     *  it will be set to channel 2, and a volume level of 5.
     */
    Television() : powerOn(false), muted(false), volume(5),
                   channel(2), prevChan(2) { }

    /** Toggles the power setting.
     *
     *  If Television is off, turns it on.
     *  If Television is on, turns it off.
     */
    togglePower() { powerOn = !powerOn; }

    /** Toggles the setting for mute.
     *
     *  If power is off, there is no effect.
     *
     *  Otherwise, if television was unmuted, it becomes muted.
     *  If television was muted, it becomes unmuted and the volume is
     *  restored to its previous setting.
     */
    void toggleMute()  {
        if (powerOn)
            muted = !muted;
    }

    /** Increments the volume of the Television by one increment.
     *
     *  If power is currently off, there is no effect (-1 returned).
     *  Otherwise, updates the volume setting appropriately.
     *
     *  If volume was at maximum level, it remains at maximum level.
     *  If television is currently muted, it will be unmuted as a result.
     *
     *  @return  the resulting volume level
     */
    int volumeUp()  {
        if (powerOn) {
            if (volume < MAX_VOLUME)
                volume++;
            muted = false;
            return volume;
        } else
            return -1;
    }

    /** Decrements the volume of the Television by one increment.
     *
     *  If power is currently off, there is no effect (-1 returned).
     *  Otherwise, updates the volume setting appropriately.
     *
     *  If volume was at minimum level, it remains at minimum level.
     *  If television is currently muted, it will be unmuted as a result.
     *
     *  @return  the resulting volume level
     */
    int volumeDown()  {
        if (powerOn) {
            if (volume > MIN_VOLUME)
                volume--;
            muted = false;
            return volume;
        } else
            return -1;
    }

    /** Increments the channel.
     *
     *  If power is off, there is no effect (-1 returned).
     *  Otherwise, updates the channel setting appropriately.
     *
     *  If channel had been set to the maximum of the valid range of
     *  channels, the effect will be to 'wrap' around resulting in the
     *  channel being set to the minimum channel.
     *
     *  @return The resulting channel setting
     */
    int channelUp()  {
        if (powerOn) {
            prevChan = channel;
            channel++;
            if (channel > MAX_CHANNEL)
                channel = MIN_CHANNEL;    // wrap around
            return channel;
        } else
            return -1;
    }

    /** Decrements the channel.
     *
     *  If power is off, there is no effect (-1 returned).
     *  Otherwise, updates the channel setting appropriately.
     *
     *  If channel had been set to the minimum of the valid range of
     *  channels, the effect will be to 'wrap' around resulting in the
     *  channel being set to the maximum channel.
     *
     *  @return The resulting channel setting
     */
    int channelDown()  {
        if powerOn {
            prevChan = channel;
            channel--;
            if (channel < MIN_CHANNEL)
                channel = MAX_CHANNEL;     // wrap around
            return channel;
        } else
            return -1;
    }

    /** Sets the channel to given number (if valid).
     *
     *  If power is off, there is no effect.
     *  If given number is illegal channel, no effect.
     *
     *  @param  number   the desired channel number
     *  @return  true if change was enacted; false otherwise.
     */
    bool setChannel(number)  {
        if ((powerOn) && (MIN_CHANNEL <= number) && (number <= MAX_CHANNEL)) {
            prevChan = channel;      // must record this before it is lost
            channel = number;
            return true;
        } else
            return false;
    }

    /** Changes the channel to most recent, previously viewed.
     *
     *  If power is off, there is no effect.
     *
     *  @return  the resulting channel setting
     */
    int jumpPrevChannel() const {
        if (powerOn) {
            int temp;
            temp = channel;
            channel = prevChan;
            prevChan = temp;
            return channel;
        } else
            return -1;
    }

    /* allows private access to external function */
    friend ostream& operator<<(ostream&, const Television&);
};

/*
 * Overloading the output operator.
 */
ostream& operator<<(ostream& out, const Television& tv) {
    out << "Power setting is currently       "
        << (tv.powerOn ? "true" : "false") << endl
        << "Channel setting is currently     "
        << tv.channel << endl
        << "(previous channel) is currently  "
        << tv.prevChan << endl
        << "Volume Setting is currently      "
        << tv.volume << endl
        << "Mute is currently                "
        << (tv.muted ? "true" : "false") << endl;
    return out;
}
```

```cpp

/** Sample unit test. */
int main() {

    Television sony;    // uses the DEFAULT constructor
    cout << "Newly created television:" << endl;
    cout << sony << endl << endl;

    sony.channelUp();
    cout << "After call to channelUp():" << endl;
    cout << sony << endl << endl;

    sony.togglePower();
    cout << "After call to togglePower():" << endl;
    cout << sony << endl << endl;

    sony.setChannel(22);
    cout << "After call to setChannel(22):" << endl;
    cout << sony << endl << endl;

    sony.jumpPrevChannel();
    cout << "After call to jumpPrevChannel():" << endl;
    cout << sony << endl << endl;

    sony.jumpPrevChannel();
    cout << "After another call to jumpPrevChannel():" << endl;
    cout << sony << endl << endl;

    sony.channelUp();
    cout << "After call to channelUp():" << endl;
    cout << sony << endl << endl;

    sony.jumpPrevChannel();
    cout << "After call to jumpPrevChannel():" << endl;
    cout << sony << endl << endl;

    sony.toggleMute();
    cout << "After call to toggleMute():" << endl;
    cout << sony << endl << endl;

    sony.volumeUp();
    cout << "After call to volumeUp():" << endl;
    cout << sony << endl << endl;

    // try to max-out the volume
    for (int i=0; i<250; i++)
        sony.volumeUp();
    cout << "After 250 calls to volumeUp():" << endl;
    cout << sony << endl << endl;

    // try to wrap-around the channel
    for (int i=0; i<250; i++)
        sony.channelDown();
    cout << "After 250 calls to channelDown():" << endl;
    cout << sony << endl << endl;
}
```