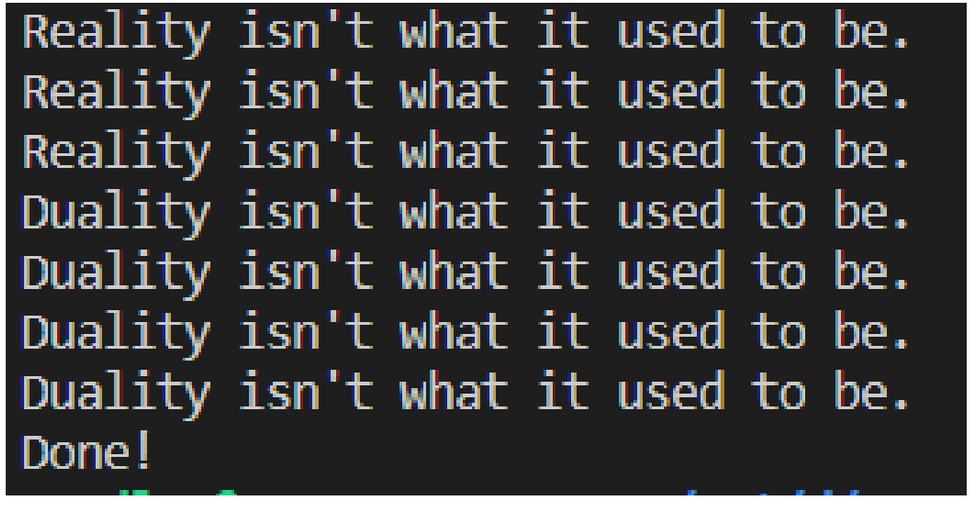
3 Exercises

1. The following is a program skeleton:

```
#include <iostream>
#include <cstring>    //for strlen(), strcpy()

struct stringy
{
    char * str;   // points to a string
    int ct;      // length of string(not counting '\0')
};

// prototypes for set() and two overloading functions show() with default arguments

int main()
{
    stringy beany;
    char testing[] = "Reality isn't what it used to be.";

    set(beany,testing);  // first argument is a reference,
        // allocates space to hold copy of testing,
        // sets str member of beany to point to the
        // new block, copies testing to the new block,
        // and sets ct member of beany

    show(beany);    //prints member string once
    show(beany, 2);  //prints member string twice

    testing[0] = 'D';
    testing[1] = 'u';
    show(testing);     //prints testing string once
    show(testing, 3);   //prints test string thrice
    show("Done!");    // prints "Done" on the screen

    // free the memory
    return 0;
}


// defines the three functions
```

Complete this skeleton by providing the described functions and prototypes. Note that there should be two **show()** functions, each using default arguments. Use **const** arguments when appropriate. Note that **set()** should use **new** to allocate sufficient space to hold the designated string.

A sample runs might look like this:



2. Write a template function **maxn()** that takes as its arguments an array of items of type T and an integer representing the number of elements in the array and that returns the largest item in the array. Test it in a program that uses the function template with an array of five int values({1,2,3,4,5}) and an array of four double values({1.1,2.7,-3.5,-2}). The program should also include a **specialization** that takes **an array of pointers-to-char** as an argument and **the number of pointers** as a second argument and that returns the address of the longest string. If multiple strings are tied for having the longest length, the function should return the address of the first one tied for longest. Test the specialization with an array of the five string pointers({“this”,”no body”,”morning”,”birds”,”sky”}).

A sample runs might look like this: