Cross Platform Escape Sequences:

Later in life you might find yourself sitting at home, drinking a mocha-flavored Soylent, and browsing through your first programs on your Apple iLens. You then realize that code might contain nasty system() calls or require windows.h or conio.h to run. Panic sets in. C has largely been replaced by Go, and Google Drive doesn't have the memories from your college years. Good times. Focus! Fear leads to desperation, and at this point you don't care if you'll have to stream the now deprecated Visual Studio Cloud. Then it hits you. Your TA wasn't like the rest, and he taught you ANSI escape sequences to keep your programs friendly to all operating systems. Yahtzee runs, and you cozy up to another caffeinated Soylent.

tl;dr: the ANSI escape sequences below allow your programs to run on most OS's.

ANSI escape sequences:

```
#define CLEAR_SCREEN "\033[2J\033[1;1H"
#define UNDER_LINE "\033[4m" //m is for SGR in ANSI
#define TEXT_RESET = "\033[0m" //MUST RESET AFTER MOST CALLS
#define STRIKE_THROUGH = "\033[9m"
#define BOLD = "\033[1m"
#define ITALICS = "\033[3m"
#define RED = "\033[31m"
#define GREEN = "\033[32m"
#define YELLOW = "\033[33m"
#define BLUE = "\033[34m"
#define MAGENTA = "\033[35m"
#define CYAN = "\033[36m"
#define WHITE = "\033[37m"
```

You will need to include windows.h in your program, but that doesn't mean you should make system() calls or use other Windows-specific functions. The way windows.h is included makes it so that it won't use windows.h on another OS. Avoid conio.h and use getchar() to pause your programs as shown here: rebrand.ly/ANSI-ed
The example below should output this:
Hello World
Hello Again
Goodbye

```c
// This goes in your header file.
#include    <stdio.h>
// Defining escape sequences allows you to use them anywhere in your program.
#define UNDER_LINE "\033[4m"
#define TEXT_RESET "\033[0m"

#ifdef _WIN32

#include    <windows.h>
#ifndef ENABLE_VIRTUAL_TERMINAL_PROCESSING
#define ENABLE_VIRTUAL_TERMINAL_PROCESSING 0x0004
#endif

#else

#define _WIN32 0
typedef enum handle { NO, YES } HANDLE;
typedef enum dword { ENABLE_VIRTUAL_TERMINAL_PROCESSING, STD_OUTPUT_HANDLE } DWORD;
HANDLE GetStdHandle( DWORD lol ) { return NO; }
void GetConsoleMode( HANDLE yah, DWORD * nah ) { return; }
void SetConsoleMode( HANDLE what, DWORD where ) { return; }

#endif

void under_line(const char* source);

int main(void) {

    // Lines 30 - 36 go in your main function.
    if (_WIN32) {
        HANDLE hOut = GetStdHandle(STD_OUTPUT_HANDLE);
        DWORD dwMode = 0;
        GetConsoleMode(hOut, &dwMode);
        dwMode |= ENABLE_VIRTUAL_TERMINAL_PROCESSING;
        SetConsoleMode(hOut, dwMode);
    }

   printf("%sHello World%s\n", UNDER_LINE, TEXT_RESET);
   under_line("Hello Again");
   printf("Goodbye\n");
   return 0; //Use this instead of exit(0);

}

// You can turn this task into a function that collects a string!
void under_line(const char* source) {
   printf("%s%s%s\n", UNDER_LINE, source, TEXT_RESET);
}
```