

CSCI 1320 HW4 Functions
(Due at the beginning of the class on Friday, March 28)

Problem: Plotting the Sine Function using functions. Call this program plot.c.

Let's develop a program that will display on the screen a plot of the values of the mathematical *sine* function (a function used in trigonometry) over some specific interval. The `sin` function takes float values (called radians, actually) and returns values between -1.0 and 1.0. It is available in the standard C library called **<math.h>**. The program will read in the interval end values `x1` and `xu`, and the increment value **delta**. Suppose, for example, that `x1` is 0.7, `xu` is 4.0, and `delta` is 0.1. That will give us a plot from 0.7 to 4.0, in increments of 0.1, of the value of the **sin** function.

The y-axis is to be displayed horizontally on the screen. The x-axis will be displayed vertically. This "sideways" orientation of the plot is much easier to produce than the more common one with the x-axis displayed vertically. For the example above, the output should look like the figure shown on the next page.

Notice that there are exactly sixty columns in the y-axis (displayed horizontally), and that the number of lines in the x-axis (displayed vertically) is given by the expression $1 + (xu - x1) / \text{delta}$, which in this case yields 34. The y-axis has a plus sign marking every unit on its scale. The leftmost, middle, and rightmost values of `y` are shown above the y-axis (-1.0, 0.0, and 1.0 in the above example). Each value of `x`, in the range from `x1` to `xu`, is displayed to the left of the x-axis, as shown above; vertical separator characters are used to draw the x-axis. Each data point is marked with an asterisk, and the entire plot is preceded by a header and followed by a footer. We begin by determining the major activities of the program and formulating them as functions that are called from the main program, shown below.

```
#include <stdio.h>
#include <math.h>

#define MAX_COLUMNS 60

float x1, xu, delta, ymin, ymax;

void print_header ( );
void get_domain_parameters ( );
void calculate_min_max ( );
void print_plot ( );

int main (void )
{
    print_header ( );
    get_domain_parameters ( );
    calculate_min_max ( );
    print_plot ( );
    return 0 ;
}
```

print_header is self-descriptive. **get_domain_parameters** gets the user input: the lower bound and upper bounds for the x-axis (`x1` and `xu`), and the interval desired on the plot (**delta**). Function **calculate_min_max** calculates the largest and smallest values of the sine function in the interval specified by the user (`ymin` and `ymax`). These values are used by other functions, so we have made them global, declared outside of main.

In this program, some variables are declared outside of all the functions. They are called **external** variables (sometimes they are also called **global** variables, as what we did in class), and they are discussed in Chapter 10. Ideally, we would like to avoid them, but a function can return only one value at a time. In a later chapter we will see how functions can return multiple values through their parameters by using pointers.

Finally, there is the **print_plot** function. It requires two functions, **print_plot_header** and **print_plot_footer**. The main loop processes values `x1` through `xu`, with an increment of **delta**, printing a line of the plot for each value. We assume **MAX_COLUMNS** is 60. In each iteration, you will calculate the proper location for the asterisk, print that location by the asterisk and print all other locations by white spaces.

I suggest placing the function in the order in which they appear above, except for the main programs, which would go first. Don't forget to add prototypes for the two new functions **print_plot_header** and **print_plot_footer**.

The **# include <math.h>** directive is required; otherwise the compiler won't know about the `sin` function. Under Linux systems, there's an additional complication: unlike the standard I/O library, the library of `math`

functions is not always automatically included during the linking phase, because of its size and fairly infrequent use. This program must then be compiled using the command

cc plot.c -lm or gcc plot.c -lm

The -lm option indicates to the linking loader that the math library is to be linked in with the programs. The -lm goes at the end of the gcc command to indicate that it is an option to the linker, not the compiler. If you compile the plot programs without the -lm option, you might get a message like this:

/tmp/cca098381.o (.text+0x3ca) : undefined reference to 'sin'

The fact that the message does not refer to a source file (but instead to /tmp), and that it mentions an "undefined reference" rather than an undeclared function, are clear indications that the problem is during linking, not compilation.

A sample run of the program is shown below. The plot covers the interval between zero and 2π ($\pi=3.141592\dots$). Notice the wave shape of the sine function; it repeats itself every interval of length 2π .

