```
#
# Makefile
#
# David J. Malan
# malan@harvard.edu
#

all: bounce button checkbox click cursor label slider text window

bounce: bounce.c Makefile
    clang -ggdb3 -O0 -std=c99 -Wall -Werror -o bounce bounce.c -lcs -lm

button: button.c Makefile
    clang -ggdb3 -O0 -std=c99 -Wall -Werror -o button button.c -lcs -lm

checkbox: checkbox.c Makefile
    clang -ggdb3 -O0 -std=c99 -Wall -Werror -o checkbox checkbox.c -lcs -lm

click: click.c Makefile
    clang -ggdb3 -O0 -std=c99 -Wall -Werror -Wno-unused-variable -o click click.c -lcs -lm

cursor: cursor.c Makefile
    clang -ggdb3 -O0 -std=c99 -Wall -Werror -o cursor cursor.c -lcs -lm

label: label.c Makefile
    clang -ggdb3 -O0 -std=c99 -Wall -Werror -o label label.c -lcs -lm

slider: slider.c Makefile
    clang -ggdb3 -O0 -std=c99 -Wall -Werror -o slider slider.c -lcs -lm

text: text.c Makefile
    clang -ggdb3 -O0 -std=c99 -Wall -Werror -o text text.c -lcs -lm

window: window.c Makefile
    clang -ggdb3 -O0 -std=c99 -Wall -Werror -o window window.c -lcs -lm

clean:
    rm -f *.o core bounce button checkbox click cursor label slider text window
```

```c
/**
 * bounce.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Bounces a circle back and forth in a window.
 */

// standard libraries
#include <stdio.h>

// Stanford Portable Library
#include <spl/gevents.h>
#include <spl/gobjects.h>
#include <spl/gwindow.h>

int main(void)
{
    // instantiate window
    GWindow window = newGWindow(320, 240);

    // instantiate circle
    GOval circle = newGOval(0, 110, 20, 20);
    setColor(circle, "BLACK");
    setFilled(circle, true);
    add(window, circle);

    // initial velocity
    double velocity = 2.0;

    // bounce forever
    while (true)
    {
        // move circle along x-axis
        move(circle, velocity, 0);

        // bounce off right edge of window
        if (getX(circle) + getWidth(circle) >= getWidth(window))
        {
            velocity = -velocity;
        }

        // bounce off left edge of window
        else if (getX(circle) <= 0)
        {
            velocity = -velocity;
        }
```

```
49.
50.         // linger before moving again
51.         pause(10);
52.     }
53. }
```

```c
/**
 * button.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Demonstrates a button.
 */

// standard libraries
#include <stdio.h>
#include <string.h>

// Stanford Portable Library
#include <spl/gevents.h>
#include <spl/ginteractors.h>
#include <spl/gwindow.h>

int main(void)
{
    // instantiate window
    GWindow window = newGWindow(320, 240);

    // instantiate button
    GButton button = newGButton("Button");
    setActionCommand(button, "click");

    // add button to southern region of window
    addToRegion(window, button, "SOUTH");

    // listen for events
    while (true)
    {
        // wait for event
        GActionEvent event = waitForEvent(ACTION_EVENT);

        // if window was closed
        if (getEventType(event) == WINDOW_CLOSED)
        {
            break;
        }

        // if action command is "click"
        if (strcmp(getActionCommand(event), "click") == 0)
        {
            printf("button was clicked\n");
        }
    }
```

```
49.
50.     // that's all folks
51.     closeGWindow(window);
52.     return 0;
53. }
```

```c
/**
 * checkbox.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Demonstrates a checkbox.
 */

// standard libraries
#include <stdio.h>
#include <string.h>

// Stanford Portable Library
#include <spl/gevents.h>
#include <spl/ginteractors.h>
#include <spl/gwindow.h>

int main(void)
{
    // instantiate window
    GWindow window = newGWindow(320, 240);

    // instantiate checkbox
    GCheckBox checkbox = newGCheckBox("I agree");
    setActionCommand(checkbox, "check");
    addToRegion(window, checkbox, "SOUTH");

    // listen for events
    while (true)
    {
        // wait for event
        GActionEvent event = waitForEvent(ACTION_EVENT);

        // if window was closed
        if (getEventType(event) == WINDOW_CLOSED)
        {
            break;
        }

        // if action command is "check"
        if (strcmp(getActionCommand(event), "check") == 0)
        {
            if (isSelected(checkbox))
            {
                printf("checkbox was checked\n");
            }
            else
```

```
49.             {
50.                 printf("checkbox was unchecked\n");
51.             }
52.         }
53.     }
54.
55.     // that's all folks
56.     closeGWindow(window);
57.     return 0;
58. }
```

```c
/**
 * click.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Prints cursor's location when mouse's button is clicked.
 */

// standard libraries
#include <stdio.h>

// Stanford Portable Library
#include <spl/gevents.h>
#include <spl/gwindow.h>

int main(void)
{
    // instantiate window
    GWindow window = newGWindow(320, 240);

    // listen forever
    while (true)
    {
        // check for mouse event
        GEvent event = getNextEvent(MOUSE_EVENT);

        // if we heard one
        if (event != NULL)
        {
            // if the event was movement
            if (getEventType(event) == MOUSE_CLICKED)
            {
                // print click's coordinates
                printf("%.0f,%.0f\n", getX(event), getY(event));
            }
        }
    }
}
```

```c
/**
 * cursor.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Draws a circle that follows user's cursor within a window.
 */

// Stanford Portable Library
#include <spl/gevents.h>
#include <spl/gobjects.h>
#include <spl/gwindow.h>

int main(void)
{
    // instantiate window
    GWindow window = newGWindow(320, 240);

    // instantiate circle
    GOval circle = newGOval(0, 0, 50, 50);

    // add circle to window
    add(window, circle);

    // follow mouse forever
    while (true)
    {
        // check for mouse event
        GEvent event = getNextEvent(MOUSE_EVENT);

        // if we heard one
        if (event != NULL)
        {
            // if the event was movement
            if (getEventType(event) == MOUSE_MOVED)
            {
                // ensure circle follows top cursor
                double x = getX(event) - getWidth(circle) / 2;
                double y = getY(event) - getWidth(circle);
                setLocation(circle, x, y);
            }
        }
    }
}
```

```c
/**
 * label.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Draws a label in a window.
 */

// standard libraries
#include <ctype.h>

// Stanford Portable Library
#include <spl/gobjects.h>
#include <spl/gwindow.h>

int main(void)
{
    // instantiate window
    GWindow window = newGWindow(320, 240);

    // instantiate label
    GLabel label = newGLabel("");
    setFont(label, "SansSerif-36");
    add(window, label);

    // count down from 50 to 0
    for (int i = 50; i >= 0; i--)
    {
        // to store 50 through 0 (with '\0'), we need <= 3 chars
        char s[3];

        // convert i from int to string
        sprintf(s, "%i", i);

        // update label
        setLabel(label, s);

        // center label
        double x = (getWidth(window) - getWidth(label)) / 2;
        double y = (getHeight(window) - getHeight(label)) / 2;
        setLocation(label, x, y);

        // linger for 100 milliseconds
        pause(100);
    }

    // that's all folks
```

```
49.     closeGWindow(window);
50.     return 0;
51. }
```

```c
/**
 * noswap.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Should swap two variables' values, but doesn't!  How come?
 */

#include <stdio.h>

void swap(int a, int b);

int main(void)
{
    int x = 1;
    int y = 2;

    printf("x is %i\n", x);
    printf("y is %i\n", y);
    printf("Swapping...\n");
    swap(x, y);
    printf("Swapped!\n");
    printf("x is %i\n", x);
    printf("y is %i\n", y);
}

/**
 * Fails to swap arguments' values.
 */
void swap(int a, int b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
```

```c
/**
 * pointers.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Prints a given string one character per line.
 *
 * Demonstrates pointer arithmetic.
 */

#include <cs50.h>
#include <stdio.h>
#include <string.h>

int main(void)
{
    // get line of text
    char* s = GetString();
    if (s == NULL)
    {
        return 1;
    }

    // print string, one character per line
    for (int i = 0, n = strlen(s); i < n; i++)
    {
        printf("%c\n", *(s+i));
    }
}
```

```c
/**
 * sigma-0.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Adds the numbers 1 through n.
 *
 * Demonstrates iteration.
 */

#include <cs50.h>
#include <stdio.h>

// prototype
int sigma(int);

int main(void)
{
    // ask user for a positive int
    int n;
        printf("Positive integer please: ");
        n = GetInt();

    // compute sum of 1 through n
    int answer = sigma(n);

    // report answer
    printf("%i\n", answer);
}

/**
 * Returns sum of 1 through m; returns 0 if m is not positive.
 */
int sigma(int m)
{
    // avoid risk of infinite loop
    // return sum of 1 through m
    int sum = 0;
    for (int i = 1; i <= m; i++)
    {
        sum += i;
    }
    return sum;
}
```

```c
/**
 * sigma-1.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Adds the numbers 1 through n.
 *
 * Demonstrates recursion.
 */

#include <cs50.h>
#include <stdio.h>

// prototype
int sigma(int);

int main(void)
{
    // ask user for a positive int
    int n;
    do
    {
        printf("Positive integer please: ");
        n = GetInt();
    }
    while (n < 1);

    // compute sum of 1 through n
    int answer = sigma(n);

    // report answer
    printf("%i\n", answer);
}

/**
 * Returns sum of 1 through m; returns 0 if m is not positive.
 */
int sigma(int m)
{
    // base case
    if (m <= 0)
    {
        return 0;
    }

    // recursive case
    else
```

```
    {
        return (m + sigma(m - 1));
    }
}
```

```c
/**
 * slider.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Demonstrates a slider.
 */

// standard libraries
#include <stdio.h>
#include <string.h>

// Stanford Portable Library
#include <spl/gevents.h>
#include <spl/ginteractors.h>
#include <spl/gwindow.h>

int main(void)
{
    // instantiate window
    GWindow window = newGWindow(320, 240);

    // instantiate slider
    addToRegion(window, newGLabel("0"), "SOUTH");
    GSlider slider = newGSlider(0, 100, 50);
    setActionCommand(slider, "slide");
    addToRegion(window, slider, "SOUTH");
    addToRegion(window, newGLabel("100"), "SOUTH");

    // listen for events
    while (true)
    {
        // wait for event
        GActionEvent event = waitForEvent(ACTION_EVENT);

        // if window was closed
        if (getEventType(event) == WINDOW_CLOSED)
        {
            break;
        }

        // if action command is "slide"
        if (strcmp(getActionCommand(event), "slide") == 0)
        {
            printf("slider was slid to %i\n", getValue(slider));
        }
    }
```

```
49.
50.     // that's all folks
51.     closeGWindow(window);
52.     return 0;
53. }
```

```c
/**
 * text.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Demonstrates a text field.
 */

// standard libraries
#include <stdio.h>
#include <string.h>

// Stanford Portable Library
#include <spl/gevents.h>
#include <spl/ginteractors.h>
#include <spl/gwindow.h>

int main(void)
{
    // instantiate window
    GWindow window = newGWindow(320, 240);

    // instantiate text field (wide enough for 10 characters)
    GTextField field = newGTextField(10);
    setActionCommand(field, "input");
    addToRegion(window, field, "SOUTH");

    // listen for events
    while (true)
    {
        // wait for event
        GActionEvent event = waitForEvent(ACTION_EVENT);

        // if window was closed
        if (getEventType(event) == WINDOW_CLOSED)
        {
            break;
        }

        // if action command is "input"
        if (strcmp(getActionCommand(event), "input") == 0)
        {
            printf("\"%s\" was inputted\n", getText(field));
        }
    }

    // that's all folks
```

```
49.     closeGWindow(window);
50.     return 0;
51. }
```

```c
/**
 * window.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Draws a window.
 */

// Stanford Portable Library
#include <spl/gwindow.h>

int main(void)
{
    // instantiate window
    GWindow window = newGWindow(320, 240);

    // let's look at it for a while
    pause(5000);

    // that's all folks
    closeGWindow(window);
    return 0;
}
```