```
#include <ncurses.h>

#include <iostream>
// - **ACS_HLINE**: Horizontal line (`─` in extended ASCII)
// - **ACS_VLINE**: Vertical line (`│` in extended ASCII)
// - **ACS_ULCORNER**: Upper left corner (`┌`)
// - **ACS_URCORNER**: Upper right corner (`┐`)
// - **ACS_LLCORNER**: Lower left corner (`└`)
// - **ACS_LRCORNER**: Lower right corner (`┘`)
// - **ACS_TTEE**: Tee pointing down (`┬`)
// - **ACS_BTEE**: Tee pointing up (`┴`)
// - **ACS_LTEE**: Tee pointing right (`├`)
// - **ACS_RTEE**: Tee pointing left (`┤`)
// - **ACS_PLUS**: Intersection (`┼`)

void draw_grid_top_line(WINDOW *win, int start_y, int start_x, int
grid_width, int cell_width) {
    // Move to the start position and draw the upper-left corner
    mvwaddch(win, start_y, start_x, ACS_ULCORNER);

    // Draw the horizontal lines and column intersections
    for (int col = 0; col < grid_width; ++col) {
        // Draw the horizontal line (length of cell_width)
        mvwhline(win, start_y, start_x + 1 + col * (cell_width + 1),
ACS_HLINE, cell_width);

        // After drawing the horizontal line, add a column intersection or
        // upper-right corner
        if (col == grid_width - 1) {
            mvwaddch(win, start_y, start_x + (col + 1) * (cell_width + 1),
ACS_URCORNER);
        } else {
            mvwaddch(win, start_y, start_x + (col + 1) * (cell_width + 1),
ACS_TTEE);
        }
    }

    // Refresh the window to show the drawn line
    wrefresh(win);
}

void draw_grid_bottom_line(WINDOW *win, int start_y, int start_x, int
grid_width, int cell_width) {
    // Move to the start position and draw the upper-left corner
    mvwaddch(win, start_y, start_x, ACS_LLCORNER);

    // Draw the horizontal lines and column intersections
    for (int col = 0; col < grid_width; ++col) {
        // Draw the horizontal line (length of cell_width)
        mvwhline(win, start_y, start_x + 1 + col * (cell_width + 1),
ACS_HLINE, cell_width);
```

```
        // After drawing the horizontal line, add a column intersection or
        // upper-right corner
        if (col == grid_width - 1) {
            mvwaddch(win, start_y, start_x + (col + 1) * (cell_width + 1),
ACS_LRCORNER);
        } else {
            mvwaddch(win, start_y, start_x + (col + 1) * (cell_width + 1),
ACS_BTEE);
        }
    }

    // Refresh the window to show the drawn line
    wrefresh(win);
}

/**
 * This function draws the bottom line of a cell in the grid. The bottom
line
 * is drawn with bottom Tees and horizontal lines. Bottom Tees are upside
down T's pointing up (`┴`)
 *
 * @param win The window to draw the lines on
 * @param start_y The starting y-coordinate of the grid
 * @param start_x The starting x-coordinate of the grid
 * @param grid_width The number of cells in the grid
 * @param cell_width The width of each cell
 *
 */
void draw_cell_bottom_line(WINDOW *win, int start_y, int start_x, int
grid_width, int cell_width) {
    // Move to the start position and draw the left-side junction (├)
    mvwaddch(win, start_y, start_x, ACS_LTEE);

    // Draw the horizontal lines and column intersections
    for (int col = 0; col < grid_width; ++col) {
        // Draw the horizontal line (length of cell_width)
        mvwhline(win, start_y, start_x + 1 + col * (cell_width + 1),
ACS_HLINE, cell_width);

        // After drawing the horizontal line, add a column intersection or
right
        // T-junction
        if (col == grid_width - 1) {
            mvwaddch(win, start_y, start_x + (col + 1) * (cell_width + 1),
ACS_RTEE);
        } else {
            mvwaddch(win, start_y, start_x + (col + 1) * (cell_width + 1),
ACS_PLUS);
        }
    }

    // Refresh the window to show the drawn line
    wrefresh(win);
}
```

```
/**
 * This function draws the vertical lines for the sides of each cell in
the grid.
 * A cell with height of two gets two vertical lines drawn.
 *
 * @param win The window to draw the lines on
 * @param start_y The starting y-coordinate of the grid
 * @param start_x The starting x-coordinate of the grid
 * @param grid_width The number of cells in the grid
 * @param cell_width The width of each cell
 * @param cell_height The height of each cell
 *
 */
void draw_cell_sides(WINDOW *win, int start_y, int start_x, int
grid_width, int cell_width, int cell_height) {
    for (int row = 0; row < cell_height; ++row) {
        // Move to the start of each row in the grid
        for (int col = 0; col <= grid_width; ++col) {
            mvwaddch(win, start_y + row, start_x + col * (cell_width + 1),
ACS_VLINE);
        }
    }
    wrefresh(win);  // Refresh to show the drawn lines
}

/**
 * This function draws a grid with the specified cell height, cell width,
grid
 * height, and grid width. The grid is drawn starting at the specified
 * start_y and start_x coordinates.
 *
 * @param start_y The starting y-coordinate of the grid
 * @param start_x The starting x-coordinate of the grid
 * @param cell_height The height of each cell
 * @param cell_width The width of each cell
 * @param grid_height The number of cells in the grid
 * @param grid_width The number of cells in the grid
 * @return void
 *
 * draw_grid(1, 5, 1, 3, 3, 3);
 */
void draw_grid(int start_y, int start_x, int cell_height, int cell_width,
int grid_height, int grid_width) {
    WINDOW *grid_win = newwin(cell_height * grid_height * 3, cell_width *
grid_width * 3, start_y, start_x);

    wattron(grid_win, COLOR_PAIR(1));  // Turn on color pair 2
    box(grid_win, 0, 0);
    wrefresh(grid_win);

    refresh();
    int curr_y = 1;
    int curr_x = 1;
    draw_grid_top_line(grid_win, 0, 0, grid_width, cell_width);
    for (int i = 0; i < grid_height; i++) {
        curr_y += 1;
        draw_cell_sides(grid_win, curr_y, start_x, grid_width, cell_width,
cell_height);
        curr_y += cell_height;
        if (i < grid_height - 1) {
            draw_cell_bottom_line(grid_win, curr_y, start_x, grid_width,
cell_width);
        }
    }
    draw_grid_bottom_line(grid_win, curr_y, start_x, grid_width,
cell_width);

    // Refresh the window to display the grid
    wrefresh(grid_win);
}

int main() {
    // Set the locale to use Unicode
    setlocale(LC_ALL, "");

    initscr();
    noecho();
    cbreak();

    // Initialize colors
    start_color();

    init_pair(1, COLOR_YELLOW, COLOR_BLACK);  // Text in red, background
black
    init_pair(2, COLOR_BLUE, COLOR_BLACK);    // Text in blue, background
black
    init_pair(3, COLOR_GREEN, COLOR_YELLOW);  // Text in blue, background
black

    // Enable Unicode support in ncurses
    if (!has_colors()) {
        endwin();
        printf("Your terminal does not support colors\n");
        return 1;
    }

    // Draw a border
    box(stdscr, 0, 0);

    // Move the cursor and print text
    attron(COLOR_PAIR(3));              // Turn on color pair 1 (red text)
    mvprintw(1, 1, "KnuckleBones!");    // Print colored text
    attroff(COLOR_PAIR(3));             // Turn off the color

    refresh();          // Ensure the message stays on screen
    curs_set(FALSE);    // Hide the cursor
    // Draw the grid @ (1, 5) with cell height 1, cell width 3, grid
height 3, and grid width 3
    draw_grid(1, 5, 1, 3, 3, 3);

    // Refresh to show changes
    refresh();

    getch();  // Wait for key press before exiting
    endwin();

    return 0;
}
```