```
  // Project 3 solution

#include "grid.h"
#include <iostream>
#include <string>
#include <cctype>
#include <cmath>
using namespace std;

const int HORIZ = 0;
const int VERT = 1;

const int FG = 0;
const int BG = 1;

const int COMMAND_OK = 0;
const int SYNTAX_ERROR = 1;
const int EXECUTION_ERROR = 2;

int performCommands(string cmd, char& plotChar, int& mode, int& badPos);
bool hasCorrectSyntax(string cmd, int& pos);
bool performOneCommand(string cmd, int& pos, int& r, int& c, char& plotChar, int& mode);
int consumeNumber(string cmd, int& pos);
bool plotLine(int r, int c, int distance, int dir, char plotChar, int fgbg);

int main()
{
    setSize(20, 30);
    char currentChar = '*';
    int currentMode = FG;
    for (;;)
    {
        cout << "Enter a command string: ";
        string cmd;
        getline(cin, cmd);
        if (cmd == "")
            break;
        int position;
        int status = performCommands(cmd, currentChar, currentMode, position);
        switch (status)
        {
          case COMMAND_OK:
            draw();
            break;
          case SYNTAX_ERROR:
            cout << "Syntax error at position " << position+1 << endl;
            break;
          case EXECUTION_ERROR:
            cout << "Cannot perform command at position " << position+1 << endl;
            break;
          default:
              // It should be impossible to get here.
            cerr << "performCommands returned " << status << "!" << endl;
        }
    }
}

int performCommands(string cmd, char& plotChar, int& mode, int& badPos)
{
    int pos;
    if ( ! hasCorrectSyntax(cmd, pos))
    {
        badPos = pos;
        return SYNTAX_ERROR;
```

```
    }

    int r = 1;
    int c = 1;
    pos = 0;

      // At the start of each loop iteration, pos is the position of the
      // start of the next plotting command within the command string, or
      // the end of that string.

    while (pos != cmd.size())
    {
        if (!performOneCommand(cmd, pos, r, c, plotChar, mode))
        {
            badPos = pos;
            return EXECUTION_ERROR;
        }
    }
    return COMMAND_OK;
}

//*********************************************************************
//  Determine whether cmd has correct syntax.
//  If it does, return true (and pos will have been modified); if not, return
//    false, and pos will be set to the position of the syntax error.
//*********************************************************************

bool hasCorrectSyntax(string cmd, int& pos)
{
    pos = 0;
    while (pos != cmd.size())
    {
        switch (toupper(cmd[pos]))
        {
          default:
            return false;
          case 'C':
            pos++;
            break;
          case 'F':
          case 'B':  // must be followd by a printable character
            pos++;
            if (pos == cmd.size()  ||  !isprint(cmd[pos]))
                return false;
            pos++;
            break;
          case 'H':
          case 'V':
            pos++;
            if (pos == cmd.size())
                return false;
            if (cmd[pos] == '-')  // optional minus sign
            {
                pos++;
                if (pos == cmd.size())
                    return false;
            }
            if (!isdigit(cmd[pos]))  // first digit
                return false;
            pos++;
            if (pos != cmd.size()  &&  isdigit(cmd[pos]))  // optional second digit
                pos++;
            break;
        }
    }
```

```
    return true;
}

//*********************************************************************
//  Execute the command starting at position pos of the cmd.  (r,c), plotChar,
//  and mode are the current position, character, and mode.
//  If successful, return true and update (r,c), plotChar, and/or mode.
//  If the command can not be performed (because it would try to plot a point
//    outside the grid), set pos to the position of the start of the command
//    and return false.
//  Precondition: cmd must be a syntactically valid command string.
//*********************************************************************

bool performOneCommand(string cmd, int& pos, int& r, int& c, char& plotChar, int& mode)
{
      // Get the character indicating the action

    char action = toupper(cmd[pos]);
    if (action == 'C')
    {
        pos++;
        clearGrid();
        r = 1;
        c = 1;
        plotChar = '*';
        mode = FG;
    }
    else if (action == 'F'  ||  action == 'B')
    {
        pos++;

          // Set the current character and the curent mode

        plotChar = cmd[pos];
        pos++;

        if (action == 'F')
            mode = FG;
        else
            mode = BG;
    }
    else if (action == 'H'  ||  action == 'V')
    {
        int commandStartPos = pos;
        pos++;

          // Get the number argument

        int distance = consumeNumber(cmd, pos);

          // Plot the line

        int dir = HORIZ;
        if (action == 'V')
            dir = VERT;
        if (!plotLine(r, c, distance, dir, plotChar, mode))
        {
            pos = commandStartPos;
            return false;
        }

          // Adjust the current position to the end of the plotted line

        if (dir == HORIZ)
            c += distance;
```

```
        else
            r += distance;
    }
    else
    {
          // It should be impossible to get here, since cmd is required to be
          // syntactically valid.
        cerr << "Plotting command action is " << action << " at position "
             << pos << "!" << endl;
        return false;  // Doesn't matter what we do here.
    }

    return true;
}

//*********************************************************************
//  Consume a number starting at position pos of the cmd.  Set pos to the
//    position just past the number and return the integer value the numeric
//    string represents.
//  Precondition:  Starting at pos, there must be an optional minus sign
//    followed by one or two digits.
//*********************************************************************

int consumeNumber(string cmd, int& pos)
{
      // Assume nonnegative to start with

    bool isNegative = false;

      // Consume optional minus sign

    if (cmd[pos] == '-')
    {
        isNegative = true;
        pos++;
    }
    int value = cmd[pos] - '0';
    pos++;

      // If there's a second digit, consume it and adjust value

    if (pos != cmd.size()  &&  isdigit(cmd[pos]))
    {
        value = 10 * value + (cmd[pos] - '0');
        pos++;
    }

      // Adjust value if there was a minus sign

    if (isNegative)
        return -value;
    else
        return value;
}

//*********************************************************************
//  plotLine
//    Using plotChar and the mode fgbg, plot a line in direction dir
//    (horizontal or vertical) starting at (r,c).  If distance is nonnegative,
//    it will extend rightward or downward; if negative, upward or leftward.
//    The other endpoint is distance units away from (r,c).  Return true if
//    successful; otherwise, plot nothing at all and return false.
//*********************************************************************

bool plotLine(int r, int c, int distance, int dir, char plotChar, int fgbg)
```

```
{
      // Validate arguments

    if (r < 1  ||  r > getRows()  ||  c < 1  ||  c > getCols())
        return false;
    switch (dir)
    {
      case HORIZ:
        if (c+distance < 1  ||  c+distance > getCols())
            return false;
        break;
      case VERT:
        if (r+distance < 1  ||  r+distance > getRows())
            return false;
        break;
      default:
        return false;
    }
    if (fgbg != FG  &&  fgbg != BG)
        return false;
    if (!isprint(plotChar))
        return false;

      // Establish the deltas for line plotting; set (dr,dc) to
      //   ( 0,  1) to plot rightward
      //   ( 0, -1) to plot leftward
      //   ( 1,  0) to plot downward
      //   (-1,  0) to plot upward
      // and set nSteps to absolute value of distance

    int dr = 0;
    int dc = 0;
    if (dir == HORIZ)
        dc = 1;
    else
        dr = 1;
    int nSteps = distance;
    if (distance < 0)
    {
        dr = -dr;
        dc = -dc;
        nSteps = -nSteps;
    }

      // Plot the line

    for (;;)
    {
          // Plot character if foreground, or if background replacing space
        if (fgbg == FG  ||  getChar(r, c) == ' ')
            setChar(r, c, plotChar);
        if (nSteps == 0)
            break;
        nSteps--;
        r += dr;
        c += dc;
    }
    return true;
}
```