This C program is a simple Graphics Description Language Interpreter. This interpreter analyzes an input text file that contains commands that produce a bitmap file. The Graphics Description Language creates a 200x200 pixel drawing canvas. A pixel can have one of the 5 possible colors depending on its value. The drawing canvas, being 200 x 200 will have coordinates ranging from (1,1) to (200,200). It should recognize the ff. commands:

P<n> x, y, col

Create point P<n> at (*x,y*) with color = *col*. Up to only 100 points (n = 1..100) can be created.

L<n> x1, y1, x2, y2, col

Create a line L<n> starting at (*x1, y1*) to (*x2,y2*) with color = *col*. Up to only 10 lines (n = 1..10) can be created.

B<n> x1, y1, x2, y2, col

Creates a box B<n> with top left corner (*x1, y1*) and bottom right corner (*x2, y2*) and filled with color = *col*. Up to only 10 boxes (n = 1..10) can be created.

C<n> *x, y, r, col*

Create a circle C<n> with center at (*x,y*) with radius *r* and filled with color =- *col*. Up to only 10 circles (n = 1..10) can be created.

MOVE *obj, x, y, col*

Moves specified object *obj* to a new center (*x, y*) and color is changed to color = *col*.

DELT *obj*

Deletes specified object *obj* from canvas.

GRAP *data_filename, col*

Creates a graph from which points are obtained from *data_filename* with color = *col*. The binary file *data_filename* should contain 200 floating point values. A black colored grid should also be drawn before drawing the points of the graph.

SAVE *bitmap_file*

Draws objects created and saves them to bitmap file *bitmap_file*.

Each object (point, circle, line, box, graph) is implemented as a structure. Each structure for a particular object will have its important information (e.g. center and radius of circle, x and y coordinate of point) along with the specific pixel value as its fields. There is an extra field that serves as a flag to know whether the structure is empty or not. This structure for the object is made to an array having length to the maximum number of objects that can be created. This array becomes a field of another structure that serves as the record book of that specific object. The graph is excluded from this additional structure since only one graph can be created.

In the command creating points, lines, boxes, and circles, a function having the string containing the command in the input text file and the structure for the specific object as its parameters is called. The function would simply fill up the specific structure array index corresponding to the object number with the information found in the string. Also, the flag is activated to note that the specific array index is occupied.

In the MOVE command, the center of the circle and the x and y coordinate of the point are changed to the point indicated in the command and the color is changed. For lines and boxes, a function is called to compute the current center of the object being moved. The difference between the two centers is then computed. This difference is then added to the point found in the structure and color is changed.

In the DELT command, the flag in the structure array index is simply deactivated to note that the index is now empty.

In the GRAP command, the binary file indicated is opened and 200 floating point values is placed to the structure along with the color value.

In the SAVE command, the BMPheader function is first called to create the bitmap file. Then the functions creating the objects in the order of the priority specified. A two-dimensional character array that has 200 rows and 200 columns is passed as a parameter in all the functions to stand for the canvas. The two indices of the array stand for the x and y coordinate of the canvas and the pixel value in that index is the color in the point in the canvas. After this, the array is read and the its values are to be written on to the bitmap file.

In creating the point, the x coordinate is the column index while the y coordinate is the row index in the character array. The value of the array in the two indices is the color. In creating the lines, the points are determined by getting the equation of the lines and then the points are drawn like what is done in creating the point. In creating a box, vertical lines are drawn until the desired box is created. In creating a circle, using the trigonometric functions, concentric circles are drawn in the character array to fill up the circle. In creating the graph, the 200 floating point values are converted to y coordinates in the canvas and then the color value placed to the character array in the corresponding indices.

In the validations needed for the interpreter, it is already assumed that the input file would have only one command per line and each command would have the exact number of operands needed for it to work. Also, the input file's operands are always separated by commas. Finally, if

the color value indicated in the command is not within the range of the pixel colors, a default value would be used.

The only errors possible to occur in the interpreter is that when the user indicated an object number not within the maximum number of objects that can be created. To check for this, an if statement is passed to check whether the value of *n* is within the object's maximum number. Also, an error would occur if any of the coordinates passed as operands in the commands result to point/s not within the canvas. This is checked by an if statement checking if x and y values are within 1 and 200. Another error would be in the Bn command that the user did not enter the top left corner and/or bottom right corner of the box he wanted to create. An if statement would check this if x1>x2 or y1<y2 where this would be invalid corners. In the Cn command, an error would occur if the radius entered is not less than 90. A simple if statement to check if the radius is less than 90 is placed to address this error. In the MOVE commands, errors would occur if the new center of points, lines, and boxes given is not within the canvas or if the new center would cause some of the object's (points, lines and boxes) to be not within the canvas. An if statement would check for the new center if it is not within the canvas. Another if statement would check if the coordinate information to replace the previous coordinate information is/are not within the canvas after taking the difference of the two center is considered. Whenever these errors would occur, an error flag is returned. This error flag would print an error specific to the flag to the screen and then the processing of the input file would immediately stop, and the program exits.