EXPERIMENT NO. 04

AIM: To implement area filling algorithm .

SOFTWARE REQUIRED: TurboC.

THEORY:

BOUNDARY FILL ALGORITHM:-

The boundary fill algorithm works as its name. This algorithm picks a point inside an object and starts to fill until it hits the boundary of the object. The color of the boundary and the color that we fill should be different for this algorithm to work.

- Start at a point inside a region
- Paint the interior outward toward the boundary
- The boundary is specified in a single color
- Fill the 4−connected or 8−connected region

```
void boundaryFill4 (int x, int y, int fill, int boundary)
{       int current; current = getPixel (x,y);
          if (current != boundary && current !=fill)
                  {         setColor(fill);
                            setPixel(x,y);
                            boundaryFill4 (x+1, y, fill, boundary);
                            boundaryFill4 (x−1, y, fill, boundary);
                            boundaryFill4(x, y+1, fill, boundary);
                             bonddaryFill4(x, y−1, fill, boundary); } }
```

FLOOD FILL ALGORITHM :-

Sometimes we come across an object where we want to fill the area and its boundary with different colors. We can paint such objects with a specified interior color instead of searching for particular boundary color as in boundary filling algorithm.

Instead of relying on the boundary of the object, it relies on the fill color. In other words, it replaces the interior color of the object with the fill color. When no more pixels of the original interior color exist, the algorithm is completed.

- Used when an area defined with multiple color boundaries
- Start at a point inside a region
- Replace a specified interior color (old color) with fill color
- Fill the 4−connected or 8−connected region until all interior points being replaced

```
void floodFill4
(int x, int y, int fill, intoldColor)
{
          if (getPixel(x,y) == oldColor)
                     {          setColor(fill);
                                setPixel(x,y);
                                floodFill4 (x+1, y, fill, oldColor);
                                floodFill4 (x−1, y, fill, oldColor);
                                floodFill4(x, y+1, fill, oldColor);
                                floodFill4(x, y−1, fill, oldColor); } }
```

CONCLUSION: Thus successfully Implemented Polygon fill algorithms.

Flood Fill Algorithm	**Boundary Fill Algorithm**
Flood fill colors an entire area in an enclosed figure through interconnected pixels using a single color	Here area gets colored with pixels of a chosen color as boundary this giving the technique its name
So, Flood Fill is one in which all connected pixels of a selected color get replaced by a fill color.	Boundary Fill is very similar with the difference being the program stopping when a given color boundary is found.
A flood fill may use an unpredictable amount of memory to finish because it isn't known how many sub-fills will be spawned	Boundary fill is usually more complicated but it is a linear algorithm and doesn't require recursion
Time Consuming	It is less time Consuming

SIGN **GRADE**

DATE