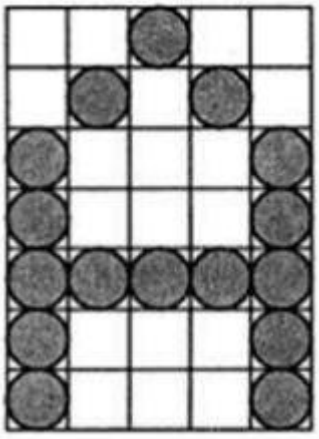
Experiment No. 9
Implement Character Generation method : Bit Map method
Name: Aryan Gaikwad
Roll Number: 09
Date of Performance:
Date of Submission:

Experiment No. 9

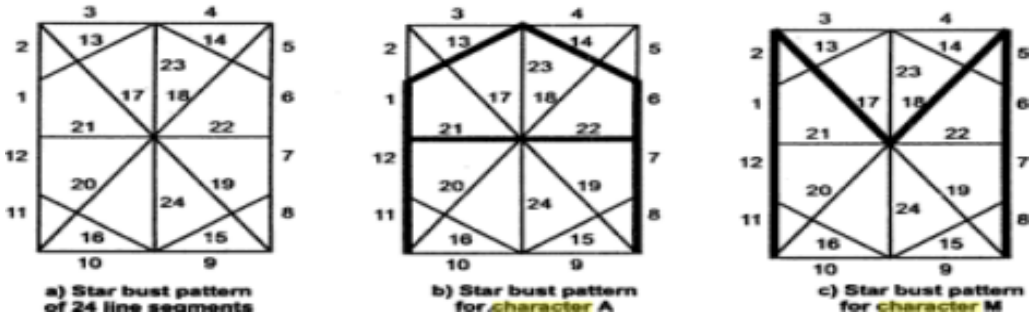
Aim: To implement Character Generation: Bit Map Method

Objective:
Identify the different Methods for Character Generation and generate the character using Stroke

Theory:
Bit map method –
Bitmap method is a called dot-matrix method as the name suggests this method use array of bits for generating a character. These dots are the points for array whose size is fixed.
· In bit matrix method when the dots are stored in the form of array the value 1 in array represent the characters i.e. where the dots appear we represent that position with numerical value 1 and the value where dots are not present is represented by 0 in array.
· It is also called dot matrix because in this method characters are represented by an array of dots in the matrix form. It is a two-dimensional array having columns and rows.
A 5x7 array is commonly used to represent characters. However, 7x9 and 9x13 arrays are also used. Higher resolution devices such as inkjet printer or laser printer may use character arrays that are over 100x100.



Starburst method –
In this method a fix pattern of line segments is used to generate characters. Out of these 24-line segments, segments required to display for particular character are highlighted. This method of character generation is called starburst method because of its characteristic appearance. The starburst patterns for characters A and M. the patterns for particular characters are stored in the form of 24 bit code, each bit representing one line segment. The bit is set to one to highlight the line segment; otherwise, it is set to zero. For example, 24-bit code for Character A is 0011 0000 0011 1100 1110 0001 and for character M is 0000 0011 0000 1100 1111 0011.



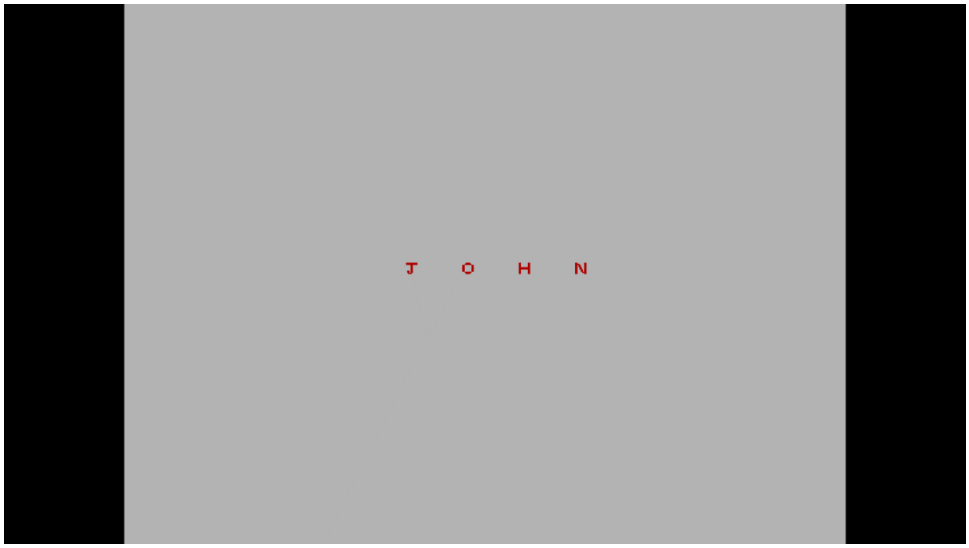
Program:

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
int main()
{
   int i,j,k,x,y;
   int gd=DETECT,gm;//DETECT is macro defined in graphics.h
   /* ch1 ch2 ch3 ch4 are character arrays that display alphabets */
   int ch1[][10]={ {1,1,1,1,1,1,1,1,1,1},
                   {1,1,1,1,1,1,1,1,1,1},
                   {0,0,0,0,1,1,0,0,0,0},
                   {0,0,0,0,1,1,0,0,0,0},
                   {0,0,0,0,1,1,0,0,0,0},
                   {0,0,0,0,1,1,0,0,0,0},
                   {0,0,0,0,1,1,0,0,0,0},
                   {0,1,1,0,1,1,0,0,0,0},
                   {0,1,1,0,1,1,0,0,0,0},
                   {0,0,1,1,1,0,0,0,0,0}};
   int ch2[][10]={ {0,0,0,1,1,1,1,0,0,0},
                   {0,0,1,1,1,1,1,1,0,0},
                   {1,1,0,0,0,0,0,0,1,1},
                   {1,1,0,0,0,0,0,0,1,1},
                   {1,1,0,0,0,0,0,0,1,1},
                   {1,1,0,0,0,0,0,0,1,1},
                   {1,1,0,0,0,0,0,0,1,1},
                   {1,1,0,0,0,0,0,0,1,1},
                   {0,0,1,1,1,1,1,1,0,0},
                   {0,0,0,1,1,1,1,0,0,0}};
   int ch3[][10]={ {1,1,0,0,0,0,0,0,1,1},
                   {1,1,0,0,0,0,0,0,1,1},
                   {1,1,0,0,0,0,0,0,1,1},
                   {1,1,0,0,0,0,0,0,1,1},
                   {1,1,1,1,1,1,1,1,1,1},
                   {1,1,1,1,1,1,1,1,1,1},
                   {1,1,0,0,0,0,0,0,1,1},
```

```
            {1,1,0,0,0,0,0,0,1,1},
            {1,1,0,0,0,0,0,0,1,1},
            {1,1,0,0,0,0,0,0,1,1}};
int ch4[][10]={ {1,1,0,0,0,0,0,0,1,1},
            {1,1,1,1,0,0,0,0,1,1},
            {1,1,0,1,1,0,0,0,1,1},
            {1,1,0,1,1,0,0,0,1,1},
            {1,1,0,0,1,1,0,0,1,1},
            {1,1,0,0,1,1,0,0,1,1},
            {1,1,0,0,0,1,1,0,1,1},
            {1,1,0,0,0,1,1,0,1,1},
            {1,1,0,0,0,0,1,1,1,1},
            {1,1,0,0,0,0,0,0,1,1}};
initgraph(&gd,&gm," ");//initialize graphic mode
setbkcolor(LIGHTGRAY);//set color of background to darkgray
for(k=0;k<4;k++)
{
   for(i=0;i<10;i++)
   {
      for(j=0;j<10;j++)
      {
         if(k==0)
         {
            if(ch1[i][j]==1)
            putpixel(j+250,i+230,RED);
         }
         if(k==1)
         {
            if(ch2[i][j]==1)
            putpixel(j+300,i+230,RED);
         }
         if(k==2)
         {
            if(ch3[i][j]==1)
            putpixel(j+350,i+230,RED);
         }
         if(k==3)
         {
            if(ch4[i][j]==1)
            putpixel(j+400,i+230,RED);
         }
      }
      delay(200);
   }
}
getch();
closegraph();
```

}

Output -



Conclusion: Comment on

1. different methods- Bitmaps are digital images made up of pixels, with monochrome using 1 bit, grayscale employing multiple bits for gray shades, and color bitmaps using more for a broad color range. They're raster graphics with fixed resolutions, detailed but prone to pixelation when scaled. Bitmaps can be compressed without quality loss (lossless) or with some quality loss (lossy), and they are edited pixel by pixel using image editing software, providing fine-grained control over image manipulation and editing.
2. advantage of stroke method- This approach, using a pixel grid to represent characters, offers precise control over character design. Designers can customize character shape, size, and style with pixel-level accuracy. Characters are represented as a set of 1s (indicating "on" pixels) and 0s (indicating "off" pixels), enabling the creation of distinctive and unique character designs. This method is widely used in pixel art, fonts, and retro video game graphics, allowing for detailed and tailored character creation with a classic, pixelated aesthetic.
3. one limitation- The stroke method offers precise control for pixelated character design, making it ideal for a retro or pixel art aesthetic. However, it's not well-suited for smooth curves or anti-aliased fonts, limiting its applicability in modern high-resolution text rendering. Modern text rendering techniques, like vector fonts and anti-aliasing, are preferred for achieving smoother, scalable, and high-quality text in contemporary designs. The stroke method excels in preserving a classic, pixelated look but may not meet the demands of modern typography and graphic design.