Name: Yash Manohar Patade

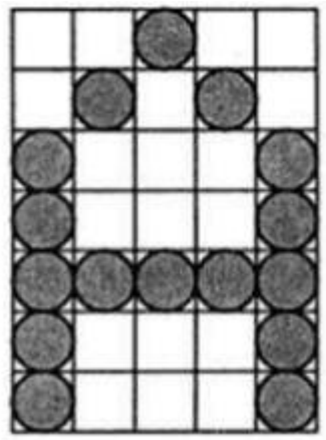
Roll No: 49

Div: 2(SE)

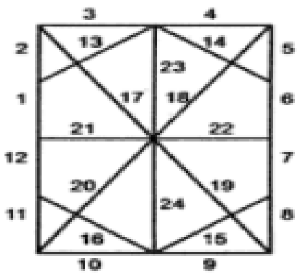
Aim: To implement Character Generation.

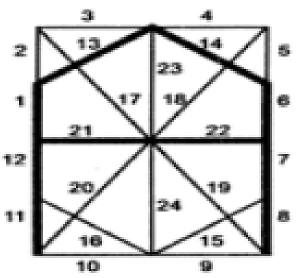
Objective:
Identify the different Methods for Character Generation and generate the character using Stroke

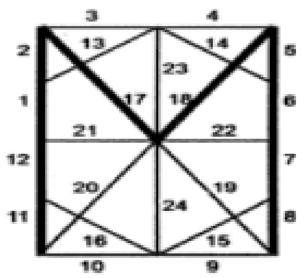
Theory:
Bit map method –
Bitmap method is a called dot-matrix method as the name suggests this method use array of bits for generating a character. These dots are the points for array whose size is fixed.
• In bit matrix method when the dots are stored in the form of array the value 1 in array represent 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.


a) Star bust pattern of 24 line segments


b) Star bust pattern for character A


c) Star bust pattern for character M

Program:

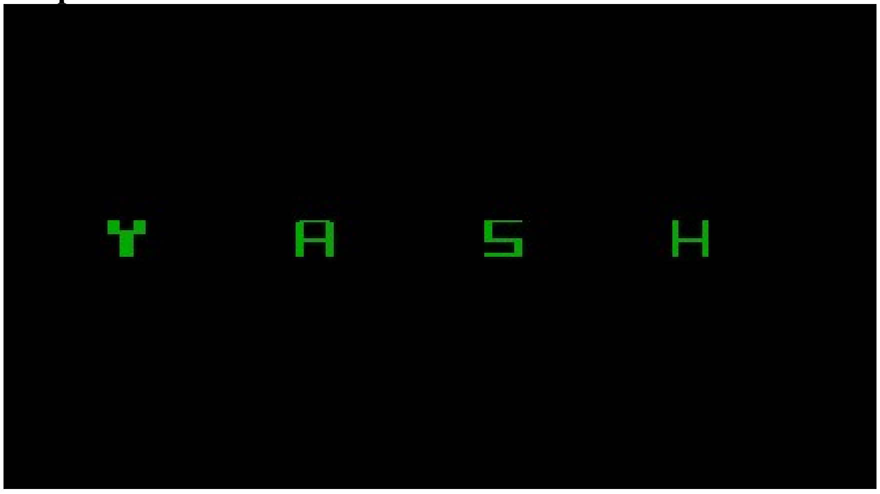
```
#include<stdio.h>

#include<conio.h>

#include<graphics.h>



void main()

{

int gd=DETECT,gm;

int i,j,k;

int a[10][10]={

                        {1,1,1,0,0,0,0,1,1,1},

                        {1,1,1,0,0,0,0,1,1,1},

                        {1,1,1,0,0,0,0,1,1,1},

                        {1,1,1,1,1,1,1,1,1,1},

                        {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},

                        {0,0,0,1,1,1,1,0,0,0},

                    };
```

```
int b[10][10]={
                                {0,1,1,1,1,1,1,1,1,0},
                                {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},
                                {1,1,1,1,1,1,1,1,1,1},
                                {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 c[10][10]={
                                {1,1,1,1,1,1,1,1,1,1},
                                {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,},
                                {1,1,1,1,1,1,1,1,1,1},
                                {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},
                                {1,1,1,1,1,1,1,1,1,1},
                               };
int d[10][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,},

                                           {1,1,0,0,0,0,0,0,1,1,},

                                           {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,},


                                          };


           initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");

           for(k=0;k<4;k++)

           {

           for(i=0;i<10;i++)

           {

           for(j=0;j<10;j++)

           {

           if(a[i][j]==1){

           putpixel(250+j,200+i,GREEN);

           }

           }

           }

           for(i=0;i<10;i++)

           {

           for(j=0;j<10;j++)

           {
```

```
if(b[i][j]==1){

putpixel(300+j,200+i,GREEN);

}

}

}


for(i=0;i<10;i++)

{

for(j=0;j<10;j++)

{

if(c[i][j]==1){

putpixel(350+j,200+i,GREEN);

}

}

}


for(i=0;i<10;i++)

{

for(j=0;j<10;j++)

{

if(d[i][j]==1){

putpixel(400+j,200+i,GREEN);

}

}

}

getch();

closegraph();
```

```
}

}
```

Output:



```
Y    A    S    H
```

Conclusion: Comment on

1. different methods
2. advantage of stroke method
3. one limitation