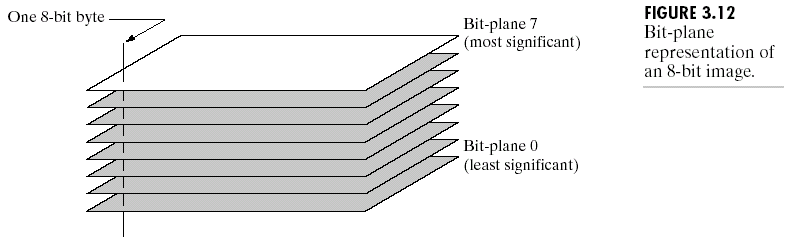
**Practical No 13.C**

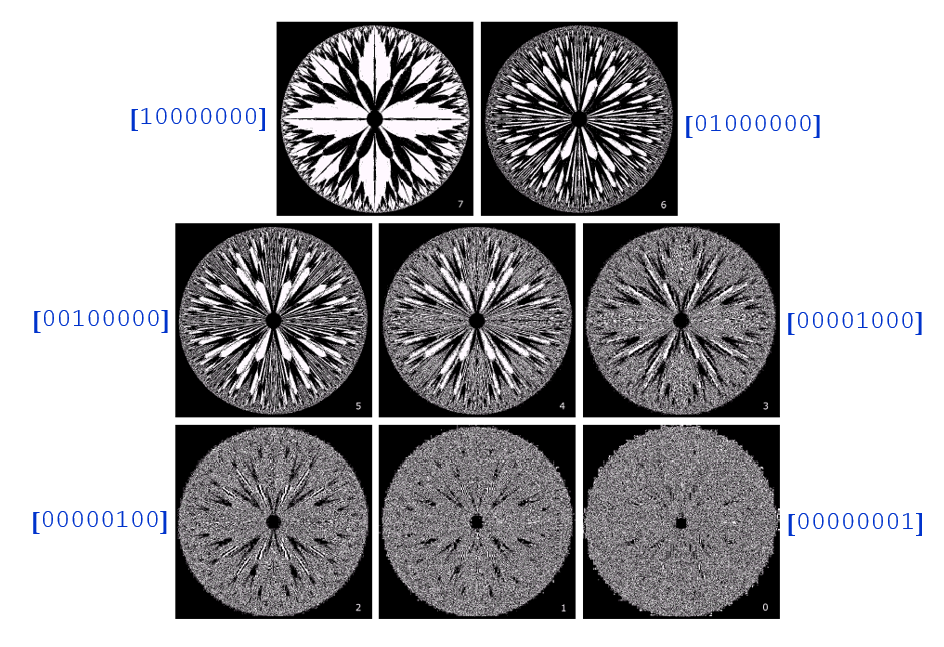
**Implementation of Piecewise-Linear Transformation Functions - Bit plane slicing.**

**Aim: Write a program to implement a Piecewise-Linear Transformation Functions using Bit plane slicing algorithm.**

**Theory:**

Instead of highlighting grey-level ranges, highlighting the contribution made to total image appearance by specific bits might be desired. Suppose that each pixel in an image is represented by 8 bits. Imagine that the image is composed of eight 1-bit planes, ranging from bit-plane 0 for the least significant bit to bit plane 7 for the most significant bit. In terms of 8-bit bytes, plane 0 contains all the lowest order bits in the bytes comprising the pixels in the image and plane 7 contains all the high-order bits.





**Conclusion: We have implemented Piecewise-Linear Transformation Functions using Bit plane slicing algorithm.**

**Code:**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<fstream.h>

#include<string.h>

#include<math.h>

struct pix {

unsigned char b,g,r; }

pixel;

int L=255;

char Header[54];

ifstream in;

ofstream out,out1;

void main() {

int gd=DETECT,gm;

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

char infile[]="C:\\TURBOC3\\BIN\\cat.bmp";

char outfile[]="C:\\TURBOC3\\BIN\\bitplane\_cat.bmp";

char imdata[]="C:\\TURBOC3\\BIN\\imdata.bmp";

in.open(infile,ios::in|ios::binary); in.read((char\*)(&Header),sizeof(Header));

out.open(outfile,ios::out|ios::binary);

out.write((char\*)(&Header),sizeof(Header));

out1.write(imdata,ios::out);

int bi;

do {

cout<<"Which bit image you want to extract 0=LSB and 7=MSB: ";

cin>>bi; }

while(bi<0||bi>7);

while(!in.eof()) {

in.read((char\*)(&pixel),sizeof(pixel));

out1<<"original"<<(int)pixel.r<<","<<(int)pixel.g<<","<<(int)pixel.b<<endl;

pixel.r=pixel.r & (int)pow(2.0,bi);

pixel.b=pixel.b & (int)pow(2.0,bi);

pixel.g=pixel.g & (int)pow(2.0,bi);

out.write((char\*)(&pixel),sizeof(pixel));

out1<<"mod"<<(int)pixel.r<<","<<(int)pixel.g<<","<<(int)pixel.b<<endl; }

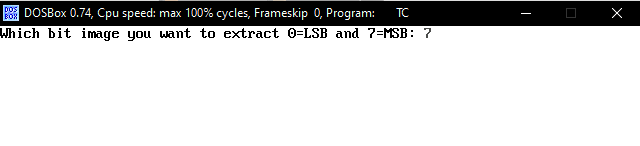
in.close();

out.close();

getch();

}

**Output:**



Original Image New Image