MATLAB Code

1. GLCM feature extraction

```
function [out] = GLCM_Features1(glcmin,pairs,a)
if ((nargin > 3) || (nargin == 0))
   error('Too many or too few input arguments. Enter GLCM and pairs.');
elseif ( (nargin == 3) )
    if ((size(glcmin,1) <= 1) || (size(glcmin,2) <= 1))
       error('The GLCM should be a 2-D or 3-D matrix.');
    elseif ( size(glcmin,1) ~= size(glcmin,2) )
        error('Each GLCM should be square with NumLevels rows and NumLevels
cols');
    end
elseif (nargin == 1) % only GLCM is entered
    pairs = 0; % default is numbers and input 1 for percentage
    if ((size(glcmin,1) <= 1) || (size(glcmin,2) <= 1))
       error('The GLCM should be a 2-D or 3-D matrix.');
    elseif ( size(glcmin,1) ~= size(glcmin,2) )
       error('Each GLCM should be square with NumLevels rows and NumLevels
cols');
    end
end


format long e
if (pairs == 1)
    newn = 1;
    for nglcm = 1:2:size(glcmin,3)
        glcm(:,:,newn)  = glcmin(:,:,nglcm) + glcmin(:,:,nglcm+1);
        newn = newn + 1;
    end
elseif (pairs == 0)
    glcm = glcmin;
end

size_glcm_1 = size(glcm,1);
size_glcm_2 = size(glcm,2);
size_glcm_3 = size(glcm,3);

% checked
out.autoc = zeros(1,size_glcm_3); % Autocorrelation: [2]
out.contr = zeros(1,size_glcm_3); % Contrast: matlab/[1,2]
out.corrm = zeros(1,size_glcm_3); % Correlation: matlab
out.corrp = zeros(1,size_glcm_3); % Correlation: [1,2]
out.cprom = zeros(1,size_glcm_3); % Cluster Prominence: [2]
out.cshad = zeros(1,size_glcm_3); % Cluster Shade: [2]
out.dissi = zeros(1,size_glcm_3); % Dissimilarity: [2]
out.energ = zeros(1,size_glcm_3); % Energy: matlab / [1,2]
out.entro = zeros(1,size_glcm_3); % Entropy: [2]
out.homom = zeros(1,size_glcm_3); % Homogeneity: matlab
out.homop = zeros(1,size_glcm_3); % Homogeneity: [2]
out.maxpr = zeros(1,size_glcm_3); % Maximum probability: [2]

out.sosvh = zeros(1,size_glcm_3); % Sum of sqaures: Variance [1]
```

```
out.savgh = zeros(1,size_glcm_3); % Sum average [1]
out.svarh = zeros(1,size_glcm_3); % Sum variance [1]
out.senth = zeros(1,size_glcm_3); % Sum entropy [1]
out.dvarh = zeros(1,size_glcm_3); % Difference variance [4]
%out.dvarh2 = zeros(1,size_glcm_3); % Difference variance [1]
out.denth = zeros(1,size_glcm_3); % Difference entropy [1]
out.inf1h = zeros(1,size_glcm_3); % Information measure of correlation1 [1]
out.inf2h = zeros(1,size_glcm_3); % Informaiton measure of correlation2 [1]
%out.mxcch = zeros(1,size_glcm_3);% maximal correlation coefficient [1]
%out.invdc = zeros(1,size_glcm_3);% Inverse difference (INV) is homom [3]
out.indnc = zeros(1,size_glcm_3); % Inverse difference normalized (INN) [3]
out.idmnc = zeros(1,size_glcm_3); % Inverse difference moment normalized [3]

% correlation with alternate definition of u and s
%out.corrm2 = zeros(1,size_glcm_3); % Correlation: matlab
%out.corrp2 = zeros(1,size_glcm_3); % Correlation: [1,2]

glcm_sum  = zeros(size_glcm_3,1);
glcm_mean = zeros(size_glcm_3,1);
glcm_var  = zeros(size_glcm_3,1);


u_x = zeros(size_glcm_3,1);
u_y = zeros(size_glcm_3,1);
s_x = zeros(size_glcm_3,1);
s_y = zeros(size_glcm_3,1);


% checked p_x p_y p_xplusy p_xminusy
p_x = zeros(size_glcm_1,size_glcm_3); % Ng x #glcms[1]
p_y = zeros(size_glcm_2,size_glcm_3); % Ng x #glcms[1]
p_xplusy = zeros((size_glcm_1*2 - 1),size_glcm_3); %[1]
p_xminusy = zeros((size_glcm_1),size_glcm_3); %[1]
% checked hxy hxy1 hxy2 hx hy
hxy  = zeros(size_glcm_3,1);
hxy1 = zeros(size_glcm_3,1);
hx   = zeros(size_glcm_3,1);
hy   = zeros(size_glcm_3,1);
hxy2 = zeros(size_glcm_3,1);

%Q    = zeros(size(glcm));

for k = 1:size_glcm_3 % number glcms

    glcm_sum(k) = sum(sum(glcm(:,:,k)));
    glcm(:,:,k) = glcm(:,:,k)./glcm_sum(k); % Normalize each glcm
    glcm_mean(k) = mean2(glcm(:,:,k)); % compute mean after norm
    glcm_var(k)  = (std2(glcm(:,:,k)))^2;

    for i = 1:size_glcm_1

        for j = 1:size_glcm_2

            out.contr(k) = out.contr(k) + (abs(i - j))^2.*glcm(i,j,k);
            out.dissi(k) = out.dissi(k) + (abs(i - j)*glcm(i,j,k));
```

```
            out.energ(k) = out.energ(k) + (glcm(i,j,k).^2);
            out.entro(k) = out.entro(k) - (glcm(i,j,k)*log(glcm(i,j,k) +
eps));
            out.homom(k) = out.homom(k) + (glcm(i,j,k)/( 1 + abs(i-j) ));
            out.homop(k) = out.homop(k) + (glcm(i,j,k)/( 1 + (i - j)^2));
            % [1] explains sum of squares variance with a mean value;
            % the exact definition for mean has not been provided in
            % the reference: I use the mean of the entire normalized glcm
            out.sosvh(k) = out.sosvh(k) + glcm(i,j,k)*((i - glcm_mean(k))^2);

            %out.invdc(k) = out.homom(k);
            out.indnc(k) = out.indnc(k) + (glcm(i,j,k)/( 1 + (abs(i-
j)/size_glcm_1) ));
            out.idmnc(k) = out.idmnc(k) + (glcm(i,j,k)/( 1 + ((i -
j)/size_glcm_1)^2));
            u_x(k)          = u_x(k) + (i)*glcm(i,j,k); % changed 10/26/08
            u_y(k)          = u_y(k) + (j)*glcm(i,j,k); % changed 10/26/08
            % code requires that Nx = Ny
            % the values of the grey levels range from 1 to (Ng)
        end

    end
    out.maxpr(k) = max(max(glcm(:,:,k)));
end
% glcms have been normalized:
% The contrast has been computed for each glcm in the 3D matrix
% (tested) gives similar results to the matlab function
r=0;

for i=1:150
    for j=1:150
      if a(i,j)>150
           r=r+1;
       end

    end
end
out.ret=r;
for k = 1:size_glcm_3

    for i = 1:size_glcm_1

        for j = 1:size_glcm_2
            p_x(i,k) = p_x(i,k) + glcm(i,j,k);
            p_y(i,k) = p_y(i,k) + glcm(j,i,k); % taking i for j and j for i
            if (ismember((i + j),[2:2*size_glcm_1]))
                p_xplusy((i+j)-1,k) = p_xplusy((i+j)-1,k) + glcm(i,j,k);
            end
            if (ismember(abs(i-j),[0:(size_glcm_1-1)]))
                p_xminusy((abs(i-j))+1,k) = p_xminusy((abs(i-j))+1,k) +...
                    glcm(i,j,k);
            end
        end
    end
```

```
end

% marginal probabilities are now available [1]
% p_xminusy has +1 in index for matlab (no 0 index)
% computing sum average, sum variance and sum entropy:
for k = 1:(size_glcm_3)

    for i = 1:(2*(size_glcm_1)-1)
        out.savgh(k) = out.savgh(k) + (i+1)*p_xplusy(i,k);
        % the summation for savgh is for i from 2 to 2*Ng hence (i+1)
        out.senth(k) = out.senth(k) - (p_xplusy(i,k)*log(p_xplusy(i,k) +
eps));
    end

end
% compute sum variance with the help of sum entropy
for k = 1:(size_glcm_3)

    for i = 1:(2*(size_glcm_1)-1)
        out.svarh(k) = out.svarh(k) + (((i+1) -
out.senth(k))^2)*p_xplusy(i,k);
        % the summation for savgh is for i from 2 to 2*Ng hence (i+1)
    end

end
% compute difference variance, difference entropy,
for k = 1:size_glcm_3
% out.dvarh2(k) = var(p_xminusy(:,k));
% but using the formula in
% http://murphylab.web.cmu.edu/publications/boland/boland_node26.html
% we have for dvarh
    for i = 0:(size_glcm_1-1)
        out.denth(k) = out.denth(k) - (p_xminusy(i+1,k)*log(p_xminusy(i+1,k)
+ eps));
        out.dvarh(k) = out.dvarh(k) + (i^2)*p_xminusy(i+1,k);
    end
end

% compute information measure of correlation(1,2) [1]
for k = 1:size_glcm_3
    hxy(k) = out.entro(k);
    for i = 1:size_glcm_1

        for j = 1:size_glcm_2
            hxy1(k) = hxy1(k) - (glcm(i,j,k)*log(p_x(i,k)*p_y(j,k) + eps));
            hxy2(k) = hxy2(k) - (p_x(i,k)*p_y(j,k)*log(p_x(i,k)*p_y(j,k) +
eps));
%             for Qind = 1:(size_glcm_1)
%                 Q(i,j,k) = Q(i,j,k) +...
%                     ( glcm(i,Qind,k)*glcm(j,Qind,k) /
(p_x(i,k)*p_y(Qind,k)) );
%             end
        end
        hx(k) = hx(k) - (p_x(i,k)*log(p_x(i,k) + eps));
```

```
        hy(k) = hy(k) - (p_y(i,k)*log(p_y(i,k) + eps));
    end
    out.inf1h(k) = ( hxy(k) - hxy1(k) ) / ( max([hx(k),hy(k)]) );
    out.inf2h(k) = ( 1 - exp( -2*( hxy2(k) - hxy(k) ) ) )^0.5;
%     eig_Q(k,:)   = eig(Q(:,:,k));
%     sort_eig(k,:)= sort(eig_Q(k,:),'descend');
%     out.mxcch(k) = sort_eig(k,2)^0.5;
% The maximal correlation coefficient was not calculated due to
% computational instability
% http://murphylab.web.cmu.edu/publications/boland/boland_node26.html
end

corm = zeros(size_glcm_3,1);
corp = zeros(size_glcm_3,1);
% using http://www.fp.ucalgary.ca/mhallbey/glcm_variance.htm for s_x s_y
for k = 1:size_glcm_3
    for i = 1:size_glcm_1
        for j = 1:size_glcm_2
            s_x(k)  = s_x(k)  + (((i) - u_x(k))^2)*glcm(i,j,k);
            s_y(k)  = s_y(k)  + (((j) - u_y(k))^2)*glcm(i,j,k);
            corp(k) = corp(k) + ((i)*(j)*glcm(i,j,k));
            corm(k) = corm(k) + (((i) - u_x(k))*((j) - u_y(k))*glcm(i,j,k));
            out.cprom(k) = out.cprom(k) + (((i + j - u_x(k) - u_y(k))^4)*...
                glcm(i,j,k));
            out.cshad(k) = out.cshad(k) + (((i + j - u_x(k) - u_y(k))^3)*...
                glcm(i,j,k));
        end
    end
    s_x(k) = s_x(k) ^ 0.5;
    s_y(k) = s_y(k) ^ 0.5;
    out.autoc(k) = corp(k);
    out.corrp(k) = (corp(k) - u_x(k)*u_y(k))/(s_x(k)*s_y(k));
    out.corrm(k) = corm(k) / (s_x(k)*s_y(k));
%     % alternate values of u and s
%     out.corrp2(k) = (corp(k) - u_x2(k)*u_y2(k))/(s_x2(k)*s_y2(k));
%     out.corrm2(k) = corm(k) / (s_x2(k)*s_y2(k));
end
```

2. K-Means classification

```
function [mu,mask]=kmeans(ima,k)

% check image
ima=double(ima);
copy=ima;          % make a copy
ima=ima(:);        % vectorize ima
mi=min(ima);       % deal with negative
ima=ima-mi+1;      % and zero values

s=length(ima);

% create image histogram

m=max(ima)+1;
h=zeros(1,m);
```

```
hc=zeros(1,m);

for i=1:s
  if(ima(i)>0) h(ima(i))=h(ima(i))+1;end;
end
ind=find(h);
hl=length(ind);

% initiate centroids

mu=(1:k)*m/(k+1);

% start process

while(true)

  oldmu=mu;
  % current classification

  for i=1:hl
      c=abs(ind(i)-mu);
      cc=find(c==min(c));
      hc(ind(i))=cc(1);
  end

  %recalculation of means

  for i=1:k,
      a=find(hc==i);
      mu(i)=sum(a.*h(a))/sum(h(a));
  end

  if(mu==oldmu) break;end;

end

% calculate mask
s=size(copy);
mask=zeros(s);
for i=1:s(1),
for j=1:s(2),
  c=abs(copy(i,j)-mu);
  a=find(c==min(c));
  mask(i,j)=a(1);
end
end

mu=mu+mi-1;    % recover real range
```

3. Feature Extraction code:

```
clc;
clear all;
close all;
```

```
[m,n]=uigetfile('*.jpg;*.tif;*.png;*.jpeg;*.bmp;*.pgm;*.jfif;*.gif','pick an
imgae');
a=imread([n,m]);
figure,imshow(a),title('Noisy image');
b=imresize(a,[256 256]);
th='s';
b=double(b)+40*rand(size(b));
[a1,b1,c1,d1]=dwt2(b,'db2');
[J,b2,c2,d2]=dwt2(a1,'db2');
[a3,b3,c3,d3]=dwt2(J,'db2');
v1=(median(abs(b3(:)))/0.6745);
b3= wthresh(b3,th,v1);
v2=(median(abs(c3(:)))/0.6745);
c3= wthresh(c3,th,v2);
v3=(median(abs(d3(:)))/0.6745);
d3= wthresh(d3,th,v3);
v1=(median(abs(b2(:)))/0.6745);
b2= wthresh(b2,th,v1);
v2=(median(abs(c2(:)))/0.6745);
c2= wthresh(c2,th,v2);
v3=(median(abs(d2(:)))/0.6745);
d2= wthresh(d2,th,v3);
v1=(median(abs(b1(:)))/0.6745);
b1= wthresh(b1,th,v1);
v2=(median(abs(c1(:)))/0.6745);
c1= wthresh(c1,th,v2);
v3=(median(abs(d1(:)))/0.6745);
d1= wthresh(d1,th,v3);
J=idwt2(a3,b3,c3,d3,'db2');
a1=idwt2(J,b2,c2,d2,'db2');
c=idwt2(a1(1:129,1:129),b1,c1,d1,'db2');
figure,imshow(c,[]),title('Denoised image');
e = roipoly(a);
ROI=zeros(400,400);
NONROI=zeros(400,400);
int16(i);
int16(j);
for i=1:150
for j=1:150
if e(i,j)==1
ROI(i,j)=a(i,j);
else
NONROI(i,j)=a(i,j);
end
end
end
figure,imshow(ROI,[]),title('Segmented image');
  k=3;
[mu,mask]=kmeans(ROI,k);
I=mask==1;
offsets = [0 1; -1 1;-1 0;-1 -1];
states = graycomatrix(ROI,'Offset',offsets);
states1=states(:,:,1);
[out] = GLCM_Features1(states1,0,a);
clas=out.ret;
%entropy=(out.entro)
%entropy1=round(entropy)
```

```
%entropy2=dec2bin(entropy1)
sumaverage=out.savgh
%sumaverage1=round(sumaverage)
%sumaverage2=dec2bin(sumaverage1)
ClusterShade=out.cshad
%ClusterShade1=round(ClusterShade)
%ClusterShade2=dec2bin(ClusterShade1)
%Maximumprobability=out.maxpr
%Maximumprobability1=round(Maximumprobability)
%Maximumprobability2=dec2bin(Maximumprobability1)
Sumofsquares=out.sosvh
%Sumofsquares1=round(Sumofsquares)
%Sumofsquares2=dec2bin(Sumofsquares1)
glcms = graycomatrix(ROI);
stats = graycoprops(glcms,'Contrast Correlation Energy Homogeneity');
Contrast = stats.Contrast
%Contrast1=round(Contrast)
%Contrast2=dec2bin(Contrast1)
Correlation = stats.Correlation
%Correlation1=round(Correlation)
%Correlation2=dec2bin(Correlation1)
Energy = stats.Energy
%Energy1=round(Energy)
%Energy2=dec2bin(Energy1)
%Homogeneity = stats.Homogeneity
%Homogeneity1=round(Homogeneity)
%Homogeneity2=dec2bin(Homogeneity1)
Mean = mean2(ROI)
%Mean1=round(Mean)
%Mean2=dec2bin(Mean1)
Standard_Deviation = std2(ROI)
%Standard_Deviation1=round(Standard_Deviation)
%Standard_Deviation2=dec2bin(Standard_Deviation1)
%Variance = mean2(var(double(ROI)))
%Variance1=round(Variance)
%Variance2=dec2bin(Variance1)
%sum_average = sum(double(ROI(:)))
%sum_average1=round(sum_average)
%sum_average2=dec2bin(sum_average1);
%Smoothness = 1-(1/(1+a));
Kurtosis = kurtosis(double(ROI(:)))
%Kurtosis1=round(Kurtosis)
%Kurtosis2=dec2bin(Kurtosis1)
Skewness = skewness(double(ROI(:)))
%Skewness1=round(Skewness)
%Skewness2=dec2bin(Skewness1)
 load Train.mat;
 nTrees=20;
features =unnamed1(1:3);
classLables = unnamed1(:,2);
  for e=1:1:16.

if unnamed1(e,1)==clas
d=1;msgbox(' cyst ');
end
if unnamed1(e,2)==clas
d=1;msgbox('Normal ');
```

```
end
if unnamed1(e,3)==clas
d=1;msgbox(' Stone ');
end
 end
```

Verilog Code

```
%%2's complement
module twocomp1(x,a,out);

input [7:0]a,x;

output [7:0]out;

wire [7:0]b;

wire [7:0]c;

assign c=~a;

assign b=8'b00000001;

assign y=c+b;

assign out=x+y;

endmodule

%%AND gate

module andg(a,b,c);

input a,b;

output c;

assign c=(a&b);

endmodule

%% Average

module refaverage255 (avg,
a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,a1,b1,c1,d1,e1,f1,g1,h1,i1,j1,k1,l1,m1,n1,o1,p1,q1,r1,s1,t1,
u1,v1,w1,x1,y1,z1,

a2,b2,c2,d2,e2,f2,g2,h2,i2,j2,k2,l2,m2,n2,o2,p2,q2,r2,s2,t2,u2,v2,w2,x2,y2,z2,a3,b3,c3,d3,e3,f3,g3,h3,i3,
j3,k3,l3,m3,n3,o3,p3,q3,r3,s3,t3,u3,v3,w3,x3,y3,z3,
```

```
a4,b4,c4,d4,e4,f4,g4,h4,i4,j4,k4,l4,m4,n4,o4,p4,q4,r4,s4,t4,u4,v4,w4,x4,y4,z4,a5,b5,c5,d5,e5,f5,g5,h5,i5,
j5,k5,l5,m5,n5,o5,p5,q5,r5,s5,t5,u5,v5,w5,x5,y5,z5,

a6,b6,c6,d6,e6,f6,g6,h6,i6,j6,k6,l6,m6,n6,o6,p6,q6,r6,s6,t6,u6,v6,w6,x6,y6,z6,a7,b7,c7,d7,e7,f7,g7,h7,i7,
j7,k7,l7,m7,n7,o7,p7,q7,r7,s7,t7,u7,v7,w7,x7,y7,z7,

a8,b8,c8,d8,e8,f8,g8,h8,i8,j8,k8,l8,m8,n8,o8,p8,q8,r8,s8,t8,u8,v8,w8,x8,y8,z8,a9,b9,c9,d9,e9,f9,g9,h9,i9,
j9,k9,l9,m9,n9,o9,p9,q9,r9,s9,t9,u9,v9);

output reg [7:0]  avg;

input [7:0] a, b
,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,a1,b1,c1,d1,e1,f1,g1,h1,i1,j1,k1,l1,m1,n1,o1,p1,q1,r1,s1,t1,u1,
v1,w1,x1,y1,z1,

a2,b2,c2,d2,e2,f2,g2,h2,i2,j2,k2,l2,m2,n2,o2,p2,q2,r2,s2,t2,u2,v2,w2,x2,y2,z2,a3,b3,c3,d3,e3,f3,g3,h3,i3,
j3,k3,l3,m3,n3,o3,p3,q3,r3,s3,t3,u3,v3,w3,x3,y3,z3,

a4,b4,c4,d4,e4,f4,g4,h4,i4,j4,k4,l4,m4,n4,o4,p4,q4,r4,s4,t4,u4,v4,w4,x4,y4,z4,a5,b5,c5,d5,e5,f5,g5,h5,i5,
j5,k5,l5,m5,n5,o5,p5,q5,r5,s5,t5,u5,v5,w5,x5,y5,z5,

a6,b6,c6,d6,e6,f6,g6,h6,i6,j6,k6,l6,m6,n6,o6,p6,q6,r6,s6,t6,u6,v6,w6,x6,y6,z6,a7,b7,c7,d7,e7,f7,g7,h7,i7,
j7,k7,l7,m7,n7,o7,p7,q7,r7,s7,t7,u7,v7,w7,x7,y7,z7,

a8,b8,c8,d8,e8,f8,g8,h8,i8,j8,k8,l8,m8,n8,o8,p8,q8,r8,s8,t8,u8,v8,w8,x8,y8,z8,a9,b9,c9,d9,e9,f9,g9,h9,i9,
j9,k9,l9,m9,n9,o9,p9,q9,r9,s9,t9,u9,v9;

reg [8:0] tt;

always @(*)

begin

  tt = a + b + c + d + e + f + g + h + i +j + k + l+ m + n + o+ p + q + r + s + t + u + v + w + x + y + z +

  a1 + b1 + c1 + d1 + e1 + f1 + g1 + h1 + i1 +j1 + k1 + l1 + m1 + n1 + o1 + p1 + q1 + r1 + s1 + t1 + u1 + v1 +
w1 + x1 + y1 + z1 +

  a2 + b2 + c2 + d2 + e2 + f2 + g2 + h2 + i2 +j2 + k2 + l2 + m2 + n2 + o2 + p2 + q2 + r2 + s2 + t2 + u2 + v2 +
w2 + x2 + y2 + z2 +

  a3 + b3 + c3 + d3 + e3 + f3 + g3 + h3 + i3 +j3 + k3 + l3 + m3 + n3 + o3 + p3 + q3 + r3 + s3 + t3 + u3 + v3 +
w3 + x3 + y3 + z3 +

  a4 + b4 + c4 + d4 + e4 + f4 + g4 + h4 + i4 +j4 + k4 + l4 + m4 + n4 + o4 + p4 + q4 + r4 + s4 + t4 + u4 + v4 +
w4 + x4 + y4 + z4 +

  a5 + b5 + c5 + d5 + e5 + f5 + g5 + h5 + i5 +j5 + k5 + l5 + m5 + n5 + o5 + p5 + q5 + r5 + s5 + t5 + u5 + v5 +
w5 + x5 + y5 + z5 +

  a6 + b6 + c6 + d6 + e6 + f6 + g6 + h6 + i6 +j6 + k6 + l6 + m6 + n6 + o6 + p6 + q6 + r6 + s6 + t6 + u6 + v6 +
w6 + x6 + y6 + z6 +
```

```
  a7 + b7 + c7 + d7 + e7 + f7 + g7 + h7 + i7 +j7 + k7 + l7 + m7 + n7 + o7 + p7 + q7 + r7 + s7 + t7 + u7 + v7 +
w7 + x7 + y7 + z7 +

  a8 + b8 + c8 + d8 + e8 + f8 + g8 + h8 + i8 +j8 + k8 + l8 + m8 + n8 + o8 + p8 + q8 + r8 + s8 + t8 + u8 + v8 +
w8 + x8 + y8 + z8 +

  a9 + b9 + c9 + d9 + e9 + f9 + g9 + h9 + i9 +j9 + k9 + l9 + m9 + n9 + o9 + p9 + q9 + r9 + s9 + t9 + u9 + v9  +
1; // 9-bit result

  avg = t>>8;


   //$display("%d\t%d\t%d", a, b, avg);

  end

endmodule


%% Average

module refaverage16(avg, a,b,c,d,e,f,g,h);

  output reg [15:0]  avg;

input [15:0] a, b ,c,d,e,f,g,h;

reg [16:0] t;

always @(*)

begin

  t = a + b + c + d + e + f + g + h +1; // 9-bit result

  avg = t>>3;


   //$display("%d\t%d\t%d", a, b, avg);

  end

endmodule

%% Booth operation

module booth1(a,b,out);

input [7:0]a;

input [7:0]b;
```

```
output reg [7:0]out;
wire [8:0]p;
wire [7:0]c;
wire [8:0]e;
wire [8:0]d;
reg [8:0]z;
reg [8:0]w;
reg [8:0]u;
reg [8:0]v;
reg [8:0]x;
reg [8:0]y;
reg [21:0]u1;
reg [21:0]v1;
reg [21:0]w1;
reg [21:0]x1;
reg [21:0]y1;
reg [21:0]z1;
reg [21:0]out1;
 assign p={b,1'b0};
complement u2(12'b000000000010,c);
shift u3(a,d);
comp u4(d,e);
always@(*)
begin
case({p[2],p[1],p[0]})
3'b000: u=12'b000000000000;
3'b001: u=a;
3'b010: u=a;
```

```
3'b011: u=d;
3'b100: u=e;
3'b101: u=c;
3'b110: u=c;
3'b111: u=12'b000000000000;
endcase
end
always@(*)
begin
case({p[4],p[3],p[2]})
3'b000: v=12'b000000000000;
3'b001: v=a;
3'b010: v=a;
3'b011: v=d;
3'b100: v=e;
3'b101: v=c;
3'b110: v=c;
3'b111: v=12'b000000000000;
endcase
end
always@(*)
begin
case({p[6],p[5],p[4]})
3'b000: w=12'b000000000000;
3'b001: w=a;
3'b010: w=a;
3'b011: w=d;
3'b100: w=e;
```

```
3'b101: w=c;
3'b110: w=c;
3'b111: w=12'b000000000000;
endcase
end
always@(*)
begin
case({p[8],p[7],p[6]})
3'b000: x=12'b000000000000;
3'b001: x=a;
3'b010: x=a;
3'b011: x=d;
3'b100: x=e;
3'b101: x=c;
3'b110: x=c;
3'b111: x=12'b000000000000;
endcase
end
always@(*)
begin
case({p[10],p[9],p[8]})
3'b000: y=12'b000000000000;
3'b001: y=a;
3'b010: y=a;
3'b011: y=d;
3'b100: y=e;
3'b101: y=c;
3'b110: y=c;
```

```
3'b111: y=12'b000000000000;
endcase
end
always@(*)
begin
case({p[12],p[11],p[10]})
3'b000: z=12'b000000000000;
3'b001: z=a;
3'b010: z=a;
3'b011: z=d;
3'b100: z=e;
3'b101: z=c;
3'b110: z=c;
3'b111: z=12'b000000000000;
endcase
u1={10'b0000000000,u};
v1={8'b00000000,v,2'b00};
w1={6'b000000,w,4'b0000};
x1={4'b0000,x,6'b000000};
y1={2'b00,y,8'b00000000};
z1={z,10'b0000000000};
out1=u1+v1+w1+x1+y1+z1;
out=out1[21:10];
end
endmodule
%% Booth multiplier
module booth (a,b,out);
input [15:0]a;
```

```
input [15:0]b;
output reg [31:0]out;
//reg [14:0]out1;
wire [16:0]p;
wire [15:0]c;
wire [16:0]e;
wire [16:0]d;
reg [16:0]w;
reg [16:0]x;
reg [16:0]y;
reg [16:0]z;
reg [16:0]ww;
reg [16:0]xx;
reg [16:0]yy;
reg [16:0]zz;
reg [30:0]w1;
reg [30:0]x1;
reg [30:0]y1;
reg [30:0]z1;
reg [30:0]w2;
reg [30:0]x2;
reg [30:0]y2;
reg [30:0]z2;
 assign p={b,1'b0};
complement2 u1(a,c);
shift2 u2(a,d);
comp2 u3(d,e);
always@(*)
```

```
begin
case({p[2],p[1],p[0]})
3'b000: w=16'b0000000000000000;
3'b001: w=a;
3'b010: w=a;
3'b011: w=d;
3'b100: w=e;
3'b101: w=c;
3'b110: w=c;
3'b111: w=16'b0000000000000000;
endcase
end

always@(*)
begin
case({p[4],p[3],p[2]})
3'b000: x=16'b0000000000000000;
3'b001: x=a;
3'b010: x=a;
3'b011: x=d;
3'b100: x=e;
3'b101: x=c;
3'b110: x=c;
3'b111: x=16'b0000000000000000;
endcase
end
always@(*)
begin
```

```
case({p[6],p[5],p[4]})
3'b000: y=16'b0000000000000000;
 3'b001: y=a;
3'b010: y=a;
3'b011: y=d;
3'b100: y=e;
3'b101: y=c;
3'b110: y=c;
3'b111: y=16'b0000000000000000;
endcase
end
always@(*)
begin
case({p[8],p[7],p[6]})
3'b000: z=16'b0000000000000000;
 3'b001: z=a;
3'b010: z=a;
3'b011: z=d;
3'b100: z=e;
3'b101: z=c;
3'b110: z=c;
3'b111: z=16'b0000000000000000;
endcase
end
always@(*)
begin
case({p[10],p[9],p[8]})
3'b000: ww=16'b0000000000000000;
```

```
 3'b001: ww=a;
3'b010: ww=a;
3'b011: ww=d;
3'b100: ww=e;
3'b101: ww=c;
3'b110: ww=c;
3'b111: ww=16'b0000000000000000;
endcase
end
always@(*)
begin
case({p[12],p[11],p[10]})
3'b000: xx=16'b0000000000000000;
 3'b001: xx=a;
3'b010: xx=a;
3'b011: xx=d;
3'b100: xx=e;
3'b101: xx=c;
3'b110: xx=c;
3'b111: xx=16'b0000000000000000;
endcase
end
always@(*)
begin
case({p[14],p[13],p[12]})
3'b000: yy=16'b0000000000000000;
 3'b001: yy=a;
3'b010: yy=a;
```

```
3'b011: yy=d;
3'b100: yy=e;
3'b101: yy=c;
3'b110: yy=c;
3'b111: yy=16'b0000000000000000;
endcase
end
always@(*)
begin
case({p[16],p[15],p[14]})
3'b000: zz=16'b0000000000000000;
 3'b001: zz=a;
3'b010: zz=a;
3'b011: zz=d;
3'b100: zz=e;
3'b101: zz=c;
3'b110: zz=c;
3'b111: zz=16'b0000000000000000;
endcase
w1={14'b000000,w};
x1={12'b0000,x,2'b00};
y1={10'b00,y,4'b0000};
z1={8'b00000000,z,6'b000000};
w2={6'b000000,ww,8'b00000000};
x2={4'b0000,xx,10'b0000000000};
y2={2'b00,yy,12'b000000000000};
z2={zz,14'b00000000000000};
out=(w1+x1+y1+z1+w2+x2+y2+z2);
```

```
//out=out1[7:0];
end
endmodule
module complement2(a,b);
input [15:0]a;
output [15:0]b;
wire [15:0]x;
wire y;
assign y=16'b0000000000000001;
assign x=~a;
assign b=x+y;
endmodule

module comp2(a,b);
input [16:0]a;
output [16:0]b;
wire [16:0]x;
wire y;
assign y=17'b00000000000000001;
assign x=~a;
assign b=x+y;
endmodule
module shift2(a,b);
input [15:0]a;
output [16:0]b;
assign b={a,1'b0};
endmodule
module booth8(a,b,out);
```

```
input [7:0]a;
input [7:0]b;
output reg [15:0]out;
//reg [14:0]out1;
wire [8:0]p;
wire [7:0]c;
wire [8:0]e;
wire [8:0]d;
reg [8:0]w;
reg [8:0]x;
reg [8:0]y;
reg [8:0]z;
reg [14:0]w1;
reg [14:0]x1;
reg [14:0]y1;
reg [14:0]z1;
 assign p={b,1'b0};
complement u1(a,c);
shift u2(a,d);
comp u3(d,e);
always@(*)
begin
case({p[2],p[1],p[0]})
3'b000: w=8'b00000000;
3'b001: w=a;
3'b010: w=a;
3'b011: w=d;
3'b100: w=e;
```

```
3'b101: w=c;
3'b110: w=c;
3'b111: w=8'b00000000;
endcase
end
always@(*)
begin
case({p[4],p[3],p[2]})
3'b000: x=8'b00000000;
3'b001: x=a;
3'b010: x=a;
3'b011: x=d;
3'b100: x=e;
3'b101: x=c;
3'b110: x=c;
3'b111: x=8'b00000000;
endcase
end
always@(*)
begin
case({p[6],p[5],p[4]})
3'b000: y=8'b00000000;
 3'b001: y=a;
3'b010: y=a;
3'b011: y=d;
3'b100: y=e;
3'b101: y=c;
3'b110: y=c;
```

```
3'b111: y=8'b00000000;
endcase
end
always@(*)
begin
case({p[8],p[7],p[6]})
3'b000: z=8'b00000000;
3'b001: z=a;
3'b010: z=a;
3'b011: z=d;
3'b100: z=e;
3'b101: z=c;
3'b110: z=c;
3'b111: z=8'b00000000;
endcase
w1={6'b000000,w};
x1={4'b0000,x,2'b00};
y1={2'b00,y,4'b0000};
z1={z,6'b000000};
out=(w1+x1+y1+z1);
//out=out1[7:0];
end
endmodule
module complement(a,b);
input [7:0]a;
output [7:0]b;
wire [7:0]x;
wire y;
```

```
assign y=8'b00000001;
assign x=~a;
assign b=x+y;
endmodule
module comp(a,b);
input [8:0]a;
output [8:0]b;
wire [8:0]x;
wire y;
assign y=9'b000000001;
assign x=~a;
assign b=x+y;
endmodule
module shift(a,b);
input [7:0]a;
output [8:0]b;
assign b={a,1'b0};
endmodule
module complement1(a,b);
input[11:0]a;
output[11:0]b;
wire[11:0]x;
wire y;
assign y=12'b000000000001;
assign x=~a;
assign b=x+y;
endmodule
module rippe_adder(X, Y, S, w8);
```

```
 input [7:0] X, Y;// Two 4-bit inputs
 output [7:0] S;
 output w8;
 wire w1, w2, w3,w4,w5,w6,w7;
 // instantiating 4 1-bit full adders in Verilog
 fulladder u1(X[0], Y[0], 1'b0, S[0], w1);
 fulladder u2(X[1], Y[1], w1, S[1], w2);
 fulladder u3(X[2], Y[2], w2, S[2], w3);
 fulladder u4(X[3], Y[3], w3, S[3], w4);
 fulladder u5(X[4], Y[4], w4, S[4], w5);
 fulladder u6(X[5], Y[5], w5, S[5], w6);
 fulladder u7(X[6], Y[6], w6, S[6], w7);
 fulladder u8(X[7], Y[7], w7, S[7], w8);
endmodule
%% Kurtosis
module kur16(clk,clr,out);
 input clk,clr;
 output[15:0]out;
 wire ra,cin1,cin2,cin3;
 wire bin,bor1,bor2,bor3,bor4,bor5,bor6,bor7,bor8,bor9,bor10,
 bor11,bor12,bor13,bor14,bor15,bor16,bor17,bor18,bor19,bor20,
 bor21,bor22,bor23,bor24,bor25,bor26,bor27,bor28,bor29,bor30,
 bor31,bor32,bor33,bor34,bor35,bor36,bor37,bor38,bor39,bor40,bor41,
 bor42,bor43,bor44,bor45,bor46,bor47,bor48,bor49,bor50,bor51,bor52,
 bor53,bor54,bor55,bor56,bor57,bor58,bor59,bor60,
 bor61,bor62,bor63,bor64,bor65,bor66,bor67,bor68,bor69,bor70,
 bor71,bor72,bor73,bor74,bor75,bor76,bor77,bor78,bor79,bor80,
 bor81,bor82,bor83,bor84,bor85,bor86,bor87,bor88,bor89,bor90,bor91,
```

bor92,bor93,bor94,bor95,bor96,bor97,bor98,bor99,bor100,

bor101,bor102,bor103,bor104,bor105,bor106,bor107,bor108,bor109,bor110,

bor111,bor112,bor113,bor114,bor115,bor116,bor117,bor118,bor119,bor120,

bor121,bor122,bor123,bor124,bor125,bor126,bor127,bor128,bor129,bor130,

bor131,bor132,bor133,bor134,bor135,bor136,bor137,bor138,bor139,bor140,bor141,

bor142,bor143,bor144,bor145,bor146,bor147,bor148,bor149,bor150,bor151,bor152,

bor153,bor154,bor155,bor156,bor157,bor158,bor159,bor160,

bor161,bor162,bor163,bor164,bor165,bor166,bor167,bor168,bor169,bor170,

bor171,bor172,bor173,bor174,bor175,bor176,bor177,bor178,bor179,bor180,

bor181,bor182,bor183,bor184,bor185,bor186,bor187,bor188,bor189,bor190,bor191,

bor192,bor193,bor194,bor195,bor196,bor197,bor198,bor199,bor200,

bor201,bor202,bor203,bor204,bor205,bor206,bor207,bor208,bor209,bor210,

bor211,bor212,bor213,bor214,bor215,bor216,bor217,bor218,bor219,bor220,

bor221,bor222,bor223,bor224,bor225,bor226,bor227,bor228,bor229,bor230,

bor231,bor232,bor233,bor234,bor235,bor236,bor237,bor238,bor239,bor240,bor241,

bor242,bor243,bor244,bor245,bor246,bor247,bor248,bor249,bor250,bor251,bor252,

bor253,bor254,bor255,bor256;

wire [7:0]a,b,c,d,e,f,g,h,i,j,k,l,mm,n,o,p,q,r,s,t,u,v,w,y,z,a1,b1,c1,d1,e1,f1,g1,h1,i1,j1,k1,l1,

m1,n1,o1,p1,q1,r1,s1,t1,u1,v1,w1,y1,z1,a2,b2,c2,d2,e2,f2,g2,h2,i2,j2,k2,l2,m2,n2,o2,p2,q2,r2,

s2,t2,u2,v2,w2,y2,z2,a3,b3,c3,d3,e3,f3,g3,h3,i3,j3,k3,l3,m3,n3,o3,p3,q3,r3,s3,t3,u3,v3,w3,y3,z3,

a4,b4,c4,d4,e4,f4,g4,h4,i4,j4,k4,l4,m4,n4,o4,p4,q4,r4,s4,t4,u4,v4,w4,y4,z4,a5,b5,c5,d5,e5,f5,g5,

h5,i5,j5,k5,l5,m5,n5,o5,p5,q5,r5,s5,t5,u5,v5,w5,y5,z5,a6,b6,c6,d6,e6,f6,g6,h6,i6,j6,k6,l6,m6,n6,

o6,p6,q6,r6,s6,t6,u6,v6,w6,y6,z6,a7,b7,c7,d7,e7,f7,g7,h7,i7,j7,k7,l7,m7,n7,o7,p7,q7,r7,s7,t7,u7,

v7,w7,y7,z7,a8,b8,c8,d8,e8,f8,g8,h8,i8,j8,k8,l8,m8,n8,o8,p8,q8,r8,s8,t8,u8,v8,w8,y8,z8,a9,b9,c9,

d9,e9,f9,g9,h9,i9,j9,k9,l9,m9,n9,o9,p9,q9,r9,s9,t9,u9,v9,w9,y9,z9,a10,b10,c10,d10,e10,f10;

wire[15:0]out1,out3,out5,out7,out9,out11,out13,out15,out17,out19,out21,out23,out25,out27,out29,ou
t31,out33,out35,out37,out39,out41,

out43,out45,out47,out49,out51,out53,out55,out57,out59,out61,out63,out65,out67,out69,out71,out73,
out75,out77,out79,out81,out83,out85,out87,

out89,out91,out93,out95,out97,out99,out101,out103,out105,out107,out109,out111,out113,out115,ou
t117,out119,out121,out123,out125,out127,

out129,out131,out133,out135,out137,out139,out141,out143,out145,out147,out149,out151,out153,out
155,out157,out159,out161,out163,out165,out167,

out169,out171,out173,out175,out177,out179,out181,out183,out185,out187,out189,out191,out193,out
195,out197,out199,out201,out203,out205,out207,out209,out211,

out213,out215,out217,out219,out221,out223,out225,out227,out229,out231,out233,out235,out237,out
239,out241,

out243,out245,out247,out249,out251,out253,out255,out257,out259,out261,out263,out265,out267,out
269,out271,out273,out275,out277,out279,out281,out283,out285,out287,

out289,out291,out293,out295,out297,out299,out301,out303,out305,out307,out309,out311,out313,out
315,out317,out319,out321,out323,out325,out327,

out329,out331,out333,out335,out337,out339,out341,out343,out345,out347,out349,out351,out353,out
355,out357,out359,out361,out363,out365,out367,

out369,out371,out373,out375,out377,out379,out381,out383,out385,out387,out389,out391,out393,out
395,out397,out399,out401,out403,out405,out407,

out409,out411,out413,out415,out417,out419,out421,out423,out425,out427,out429,out431,out433,out
435,out437,out439,out441,out443,out445,out447,out449,out451,out453,out455,out457,out459,out46
1,out463,out465,out467,

out469,out471,out473,out475,out477,out479,out481,out483,out485,out487,out489,out491,out493,out
495,out497,out499,out501,out503,out505,out507,out509,out511,

out513;

wire[31:0]out2,out4,out6,out8,out10,out12,out14,out16,out18,out20,out22,out24,out26,out28,out30,o
ut32,out34,out36,out38,out40,out42,

out44,out46,out48,out50,out52,out54,out56,out58,out60,out62,out64,out66,out68,out70,out72,out74,
out76,out78,out80,out82,out84,out86,out88,

out90,out92,out94,out96,out98,out100,out102,out104,out106,out108,out110,out112,out114,out116,o
ut118,out120,out122,out124,out126,out128,

out130,out132,out134,out136,out138,out140,out142,out144,out146,out148,out150,out152,out154,out
156,out158,out160,out162,out164,out166,out168,

out170,out172,out174,out176,out178,out180,out182,out184,out186,out188,out190,out192,out194,out
196,out198,out200,out202,out204,out206,out208,out210,out212,

out214,out216,out218,out220,out222,out224,out226,out228,out230,out232,out234,out236,out238,out
240,out242,

out244,out246,out248,out250,out252,out254,out256,out258,out260,out262,out264,out266,out268,out
270,out272,out274,out276,out278,out280,out282,out284,out286,out288,

out290,out292,out294,out296,out298,out300,out302,out304,out306,out308,out310,out312,out314,out
316,out318,out320,out322,out324,out326,out328,

out330,out332,out334,out336,out338,out340,out342,out344,out346,out348,out350,out352,out354,out
356,out358,out360,out362,out364,out366,out368,

out370,out372,out374,out376,out378,out380,out382,out384,out386,out388,out390,out392,out394,out
396,out398,out400,out402,out404,out406,out408,

out410,out412,out414,out416,out418,out420,out422,out424,out426,out428,out430,out432,out434,out
436,out438,out440,out442,out444,out446,out448,out450,out452,out454,out456,out458,out460,out46
2,out464,out466,out468,

out470,out472,out474,out476,out478,out480,out482,out484,out486,out488,out490,out492,out494,out
496,out498,out500,out502,out504,out506,out508,out510,out512,

out514,aa,bb,cc,dd,ee,ff,gg,hh,ii,jj,kk,ll,mmm,nn,oo,pp,qq,rr,ss,tt,uu,vv,ww,xx,yy,zz,aa1,bb1,cc1,dd1,ee
1,ff1,res1,res2,res3,res4,res5,res6,res7;

reg [7:0]m[255:0];

```
 wire [3:0]la;
 assign la=4'b1000;
assign ra=1'b0;
wire [7:0]x;
assign x= 8'b00100001;
assign bin=1'b0;
initial
 begin
 $readmemb("D:/rashmi pjt/rr.txt",m);
end
//kurr d11(clka,rsta,wea,addra,dina,douta);
hs8 uu1(m[0],x,bin,a,bor1);
hs8 uu2(m[1],x,bin,b,bor2);
hs8 uu3(m[2],x,bin,c,bor3);
hs8 uu4(m[3],x,bin,d,bor4);
hs8 uu5(m[4],x,bin,e,bor5);
hs8 uu6(m[5],x,bin,f,bor6);
hs8 uu7(m[6],x,bin,g,bor7);
hs8 uu8(m[7],x,bin,h,bor8);
hs8 uu9(m[8],x,bin,i,bor9);
hs8 u10(m[9],x,bin,j,bor10);
hs8 u11(m[10],x,bin,k,bor11);
hs8 u12(m[11],x,bin,l,bor12);
hs8 u13(m[12],x,bin,mm,bor13);
hs8 u14(m[13],x,bin,n,bor14);
hs8 u15(m[14],x,bin,o,bor15);
hs8 u16(m[15],x,bin,p,bor16);
hs8 u17(m[16],x,bin,q,bor17);
```

```
hs8 u18(m[17],x,bin,r,bor18);
hs8 u19(m[18],x,bin,s,bor19);
hs8 u20(m[19],x,bin,t,bor20);
hs8 u21(m[20],x,bin,u,bor21);
hs8 u22(m[21],x,bin,v,bor22);
hs8 u23(m[22],x,bin,w,bor23);
hs8 u24(m[23],x,bin,y,bor24);
hs8 u25(m[24],x,bin,z,bor25);
hs8 u26(m[25],x,bin,a1,bor26);
hs8 u27(m[26],x,bin,b1,bor27);
hs8 u28(m[27],x,bin,c1,bor28);
hs8 u29(m[28],x,bin,d1,bor29);
hs8 u30(m[29],x,bin,e1,bor30);
hs8 u31(m[30],x,bin,f1,bor31);
hs8 u32(m[31],x,bin,g1,bor32);
hs8 u33(m[32],x,bin,h1,bor33);
hs8 u34(m[33],x,bin,i1,bor34);
hs8 u35(m[34],x,bin,j1,bor35);
hs8 u36(m[35],x,bin,k1,bor36);
hs8 u37(m[36],x,bin,l1,bor37);
hs8 u38(m[37],x,bin,m1,bor38);
hs8 u39(m[38],x,bin,n1,bor39);
hs8 u40(m[39],x,bin,o1,bor40);
hs8 u41(m[40],x,bin,p1,bor41);
hs8 u42(m[41],x,bin,q1,bor42);
hs8 u43(m[42],x,bin,r1,bor43);
hs8 u44(m[43],x,bin,s1,bor44);
hs8 u45(m[44],x,bin,t1,bor45);
```

```
hs8 u46(m[45],x,bin,u1,bor46);
hs8 u47(m[46],x,bin,v1,bor47);
hs8 u48(m[47],x,bin,w1,bor48);
hs8 u49(m[48],x,bin,y1,bor49);
hs8 u50(m[49],x,bin,z1,bor50);
hs8 u51(m[50],x,bin,a2,bor51);
hs8 u52(m[51],x,bin,b2,bor52);
hs8 u53(m[52],x,bin,c2,bor53);
hs8 u54(m[53],x,bin,d2,bor54);
hs8 u55(m[54],x,bin,e2,bor55);
hs8 u56(m[55],x,bin,f2,bor56);
hs8 u57(m[56],x,bin,g2,bor57);
hs8 u58(m[57],x,bin,h2,bor58);
hs8 u59(m[58],x,bin,i2,bor59);
hs8 u60(m[59],x,bin,j2,bor60);
hs8 u61(m[60],x,bin,k2,bor61);
hs8 u62(m[61],x,bin,l2,bor62);
hs8 u63(m[62],x,bin,m2,bor63);
hs8 u64(m[63],x,bin,n2,bor64);
hs8 u65(m[64],x,bin,o2,bor65);
hs8 u66(m[65],x,bin,p2,bor66);
hs8 u67(m[66],x,bin,q2,bor67);
hs8 u68(m[67],x,bin,r2,bor68);
hs8 u69(m[68],x,bin,s2,bor69);
hs8 u70(m[69],x,bin,t2,bor70);
hs8 u71(m[70],x,bin,u2,bor71);
hs8 u72(m[71],x,bin,v2,bor72);
hs8 u73(m[72],x,bin,w2,bor73);
```

```
hs8 u74(m[73],x,bin,y2,bor74);
hs8 u75(m[74],x,bin,z2,bor75);
hs8 u76(m[75],x,bin,a3,bor76);
hs8 u77(m[76],x,bin,b3,bor77);
hs8 u78(m[77],x,bin,c3,bor78);
hs8 u79(m[78],x,bin,d3,bor79);
hs8 u80(m[79],x,bin,e3,bor80);
hs8 u81(m[80],x,bin,f3,bor81);
hs8 u82(m[81],x,bin,g3,bor82);
hs8 u83(m[82],x,bin,h3,bor83);
hs8 u84(m[83],x,bin,i3,bor84);
hs8 u85(m[84],x,bin,j3,bor85);
hs8 u86(m[85],x,bin,k3,bor86);
hs8 u87(m[86],x,bin,l3,bor87);
hs8 u88(m[87],x,bin,m3,bor88);
hs8 u89(m[88],x,bin,n3,bor89);
hs8 u90(m[89],x,bin,o3,bor90);
hs8 u91(m[90],x,bin,p3,bor91);
hs8 u92(m[91],x,bin,q3,bor92);
hs8 u93(m[92],x,bin,r3,bor93);
hs8 u94(m[93],x,bin,s3,bor94);
hs8 u95(m[94],x,bin,t3,bor95);
hs8 u96(m[95],x,bin,u3,bor96);
hs8 u97(m[96],x,bin,v3,bor97);
hs8 u98(m[97],x,bin,w3,bor98);
hs8 u99(m[98],x,bin,y3,bor99);
hs8 u100(m[99],x,bin,z3,bor100);
hs8 u101(m[100],x,bin,a4,bor101);
```

```
hs8 u102(m[101],x,bin,b4,bor102);
hs8 u103(m[102],x,bin,c4,bor103);
hs8 u104(m[103],x,bin,d4,bor104);
hs8 u105(m[104],x,bin,e4,bor105);
hs8 u106(m[105],x,bin,f4,bor106);
hs8 u107(m[106],x,bin,g4,bor107);
hs8 u108(m[107],x,bin,h4,bor108);
hs8 u109(m[108],x,bin,i4,bor109);
hs8 u110(m[109],x,bin,j4,bor110);
hs8 u111(m[110],x,bin,k4,bor111);
hs8 u112(m[111],x,bin,l4,bor112);
hs8 u113(m[112],x,bin,m4,bor113);
hs8 u114(m[113],x,bin,n4,bor114);
hs8 u115(m[114],x,bin,o4,bor115);
hs8 u116(m[115],x,bin,p4,bor116);
hs8 u117(m[116],x,bin,q4,bor117);
hs8 u118(m[117],x,bin,r4,bor118);
hs8 u119(m[118],x,bin,s4,bor119);
hs8 u120(m[119],x,bin,t4,bor120);
hs8 u121(m[120],x,bin,u4,bor121);
hs8 u122(m[121],x,bin,v4,bor122);
hs8 u123(m[122],x,bin,w4,bor123);
hs8 u124(m[123],x,bin,y4,bor124);
hs8 u125(m[124],x,bin,z4,bor125);
hs8 u126(m[125],x,bin,a5,bor126);
hs8 u127(m[126],x,bin,b5,bor127);
hs8 u128(m[127],x,bin,c5,bor128);
hs8 u129(m[128],x,bin,d5,bor129);
```

```
hs8 u130(m[129],x,bin,e5,bor130);
hs8 u131(m[130],x,bin,f5,bor131);
hs8 u132(m[131],x,bin,g5,bor132);
hs8 u133(m[132],x,bin,h5,bor133);
hs8 u134(m[133],x,bin,i5,bor134);
hs8 u135(m[134],x,bin,j5,bor135);
hs8 u136(m[135],x,bin,k5,bor136);
hs8 u137(m[136],x,bin,l5,bor137);
hs8 u138(m[137],x,bin,m5,bor138);
hs8 u139(m[138],x,bin,n5,bor139);
hs8 u140(m[139],x,bin,o5,bor140);
hs8 u141(m[140],x,bin,p5,bor141);
hs8 u142(m[141],x,bin,q5,bor142);
hs8 u143(m[142],x,bin,r5,bor143);
hs8 u144(m[143],x,bin,s5,bor144);
hs8 u145(m[144],x,bin,t5,bor145);
hs8 u146(m[145],x,bin,u5,bor146);
hs8 u147(m[146],x,bin,v5,bor147);
hs8 u148(m[147],x,bin,w5,bor148);
hs8 u149(m[148],x,bin,y5,bor149);
hs8 u150(m[149],x,bin,z5,bor150);
hs8 u151(m[150],x,bin,a6,bor151);
hs8 u152(m[151],x,bin,b6,bor152);
hs8 u153(m[152],x,bin,c6,bor153);
hs8 u154(m[153],x,bin,d6,bor154);
hs8 u155(m[154],x,bin,e6,bor155);
hs8 u156(m[155],x,bin,f6,bor156);
hs8 u157(m[156],x,bin,g6,bor157);
```

```
hs8 u158(m[157],x,bin,h6,bor158);
hs8 u159(m[158],x,bin,i6,bor159);
hs8 u160(m[159],x,bin,j6,bor160);
hs8 u161(m[160],x,bin,k6,bor161);
hs8 u162(m[161],x,bin,l6,bor162);
hs8 u163(m[162],x,bin,m6,bor163);
hs8 u164(m[163],x,bin,n6,bor164);
hs8 u165(m[164],x,bin,o6,bor165);
hs8 u166(m[165],x,bin,p6,bor166);
hs8 u167(m[166],x,bin,q6,bor167);
hs8 u168(m[167],x,bin,r6,bor168);
hs8 u169(m[168],x,bin,s6,bor169);
hs8 u170(m[169],x,bin,t6,bor170);
hs8 u171(m[170],x,bin,u6,bor171);
hs8 u172(m[171],x,bin,v6,bor172);
hs8 u173(m[172],x,bin,w6,bor173);
hs8 u174(m[173],x,bin,y6,bor174);
hs8 u175(m[174],x,bin,z6,bor175);
hs8 u176(m[175],x,bin,a7,bor176);
hs8 u177(m[176],x,bin,b7,bor177);
hs8 u178(m[177],x,bin,c7,bor178);
hs8 u179(m[178],x,bin,d7,bor179);
hs8 u180(m[179],x,bin,e7,bor180);
hs8 u181(m[180],x,bin,f7,bor181);
hs8 u182(m[181],x,bin,g7,bor182);
hs8 u183(m[182],x,bin,h7,bor183);
hs8 u184(m[183],x,bin,i7,bor184);
hs8 u185(m[184],x,bin,j7,bor185);
```

```
hs8 u186(m[185],x,bin,k7,bor186);
hs8 u187(m[186],x,bin,l7,bor187);
hs8 u188(m[187],x,bin,m7,bor188);
hs8 u189(m[188],x,bin,n7,bor189);
hs8 u190(m[189],x,bin,o7,bor190);
hs8 u191(m[190],x,bin,p7,bor191);
hs8 u192(m[191],x,bin,q7,bor192);
hs8 u193(m[192],x,bin,r7,bor193);
hs8 u194(m[193],x,bin,s7,bor194);
hs8 u195(m[194],x,bin,t7,bor195);
hs8 u196(m[195],x,bin,u7,bor196);
hs8 u197(m[196],x,bin,v7,bor197);
hs8 u198(m[197],x,bin,w7,bor198);
hs8 u199(m[198],x,bin,y7,bor199);
hs8 u200(m[199],x,bin,z7,bor200);
hs8 u201(m[200],x,bin,a8,bor201);
hs8 u202(m[201],x,bin,b8,bor202);
hs8 u203(m[202],x,bin,c8,bor203);
hs8 u204(m[203],x,bin,d8,bor204);
hs8 u205(m[204],x,bin,e8,bor205);
hs8 u206(m[205],x,bin,f8,bor206);
hs8 u207(m[206],x,bin,g8,bor207);
hs8 u208(m[207],x,bin,h8,bor208);
hs8 u209(m[208],x,bin,i8,bor209);
hs8 u210(m[209],x,bin,j8,bor210);
hs8 u211(m[210],x,bin,k8,bor211);
hs8 u212(m[211],x,bin,l8,bor212);
hs8 u213(m[212],x,bin,m8,bor213);
```

```
hs8 u214(m[213],x,bin,n8,bor214);
hs8 u215(m[214],x,bin,o8,bor215);
hs8 u216(m[215],x,bin,p8,bor216);
hs8 u217(m[216],x,bin,q8,bor217);
hs8 u218(m[217],x,bin,r8,bor218);
hs8 u219(m[218],x,bin,s8,bor219);
hs8 u220(m[219],x,bin,t8,bor220);
hs8 u221(m[220],x,bin,u8,bor221);
hs8 u222(m[221],x,bin,v8,bor222);
hs8 u223(m[222],x,bin,w8,bor223);
hs8 u224(m[223],x,bin,y8,bor224);
hs8 u225(m[224],x,bin,z8,bor225);
hs8 u226(m[225],x,bin,a9,bor226);
hs8 u227(m[226],x,bin,b9,bor227);
hs8 u228(m[227],x,bin,c9,bor228);
hs8 u229(m[228],x,bin,d9,bor229);
hs8 u230(m[229],x,bin,e9,bor230);
hs8 u231(m[230],x,bin,f9,bor231);
hs8 u232(m[231],x,bin,g9,bor232);
hs8 u233(m[232],x,bin,h9,bor233);
hs8 u234(m[233],x,bin,i9,bor234);
hs8 u235(m[234],x,bin,j9,bor235);
hs8 u236(m[235],x,bin,k9,bor236);
hs8 u237(m[236],x,bin,l9,bor237);
hs8 u238(m[237],x,bin,m9,bor238);
hs8 u239(m[238],x,bin,n9,bor239);
hs8 u240(m[239],x,bin,o9,bor240);
hs8 u241(m[240],x,bin,p9,bor241);
```

```
hs8 u242(m[241],x,bin,q9,bor242);
hs8 u243(m[242],x,bin,r9,bor243);
hs8 u244(m[243],x,bin,s9,bor244);
hs8 u245(m[244],x,bin,t9,bor245);
hs8 u246(m[245],x,bin,u9,bor246);
hs8 u247(m[246],x,bin,v9,bor247);
hs8 u248(m[247],x,bin,w9,bor248);
hs8 u249(m[248],x,bin,y9,bor249);
hs8 u250(m[249],x,bin,z9,bor250);
hs8 u251(m[250],x,bin,a10,bor251);
hs8 u252(m[251],x,bin,b10,bor252);
hs8 u253(m[252],x,bin,c10,bor253);
hs8 u254(m[253],x,bin,d10,bor254);
hs8 u255(m[254],x,bin,e10,bor255);
hs8 u256(m[255],x,bin,f10,bor256);
booth8 vv1(a,a,out1);
booth vv2(out1,out1,out2);
booth8 vv3(b,b,out3);
booth vv4(out3,out3,out4);
booth8 vv5(c,c,out5);
booth vv6(out5,out5,out6);
booth8 vv7(d,d,out7);
booth vv8(out7,out7,out8);
booth8 vv9(e,e,out9);
booth v10(out9,out9,out10);
booth8 v11(f,f,out11);
booth v12(out11,out11,out12);
booth8 v13(g,g,out13);
```

```
booth v14(out13,out13,out14);
booth8 v15(h,h,out15);
booth v16(out15,out15,out16);
booth8 v17(i,i,out17);
booth v18(out17,out17,out18);
booth8 v19(j,j,out19);
booth v20(out19,out19,out20);
booth8 v21(k,k,out21);
booth v22(out21,out21,out22);
booth8 v23(l,l,out23);
booth v24(out23,out23,out24);
booth8 v25(mm,mm,out25);
booth v26(out25,out25,out26);
booth8 v27(n,n,out27);
booth v28(out27,out27,out28);
booth8 v29(o,o,out29);
booth v30(out29,out29,out30);
booth8 v31(p,p,out31);
booth v32(out31,out31,out32);
booth8 v33(q,q,out33);
booth v34(out33,out33,out34);
booth8 v35(r,r,out35);
booth v36(out35,out35,out36);
booth8 v37(s,s,out37);
booth v38(out37,out37,out38);
booth8 v39(t,t,out39);
booth v40(out39,out39,out40);
booth8 v41(u,u,out41);
```

```
booth v42(out41,out41,out42);
booth8 v43(v,v,out43);
booth v44(out43,out43,out44);
booth8 v45(w,w,out45);
booth v46(out45,out45,out46);
booth8 v47(y,y,out47);
booth v48(out47,out47,out48);
booth8 v49(z,z,out49);
booth v50(out49,out49,out50);
booth8 v51(a1,a1,out51);
booth v52(out51,out51,out52);
booth8 v53(b1,b1,out53);
booth v54(out53,out53,out54);
booth8 v55(c1,c1,out55);
booth v56(out55,out55,out56);
booth8 v57(d1,d1,out57);
booth v58(out57,out57,out58);
booth8 v59(e1,e1,out59);
booth v60(out59,out59,out60);
booth8 v61(f1,f1,out61);
booth v62(out61,out61,out62);
booth8 v63(g1,g1,out63);
booth v64(out63,out63,out64);
booth8 v65(h1,h1,out65);
booth v66(out65,out65,out66);
booth8 v67(i1,i1,out67);
booth v68(out67,out67,out68);
booth8 v69(j1,j1,out69);
```

```
booth v70(out69,out69,out70);
booth8 v71(k1,k1,out71);
booth v72(out71,out71,out72);
booth8 v73(l1,l1,out73);
booth v74(out73,out73,out74);
booth8 v75(m1,m1,out75);
booth v76(out75,out75,out76);
booth8 v77(n1,n1,out77);
booth v78(out77,out77,out78);
booth8 v79(o1,o1,out79);
booth v80(out79,out79,out80);
booth8 v81(p1,p1,out81);
booth v82(out81,out81,out82);
booth8 v83(q1,q1,out83);
booth v84(out83,out83,out84);
booth8 v85(r1,r1,out85);
booth v86(out85,out85,out86);
booth8 v87(s1,s1,out87);
booth v88(out87,out87,out88);
booth8 v89(t1,t1,out89);
booth v90(out89,out89,out90);
booth8 v91(u1,u1,out91);
booth v92(out91,out91,out92);
booth8 v93(v1,v1,out93);
booth v94(out93,out93,out94);
booth8 v95(w1,w1,out95);
booth v96(out95,out95,out96);
booth8 v97(y1,y1,out97);
```

```
booth v98(out97,out97,out98);
booth8 v99(z1,z1,out99);
booth v100(out99,out99,out100);
booth8 v101(a2,a2,out101);
booth v102(out101,out101,out102);
booth8 v103(b2,b2,out103);
booth v104(out103,out103,out104);
booth8 v105(c2,c2,out105);
booth v106(out105,out105,out106);
booth8 v107(d2,d2,out107);
booth v108(out107,out107,out108);
booth8 v109(e2,e2,out109);
booth v110(out109,out109,out110);
booth8 v111(f2,f2,out111);
booth v112(out111,out111,out112);
booth8 v113(g2,g2,out113);
booth v114(out113,out113,out114);
booth8 v115(h2,h2,out115);
booth v116(out115,out115,out116);
booth8 v117(i2,i2,out117);
booth v118(out117,out117,out118);
booth8 v119(j2,j2,out119);
booth v120(out119,out119,out120);
booth8 v121(k2,k2,out121);
booth v122(out121,out121,out122);
booth8 v123(l2,l2,out123);
booth v124(out123,out123,out124);
booth8 v125(m2,m2,out125);
```

```
booth v126(out125,out125,out126);
booth8 v127(n2,n2,out127);
booth v128(out127,out127,out128);
booth8 v129(o2,o2,out129);
booth v130(out129,out129,out130);
booth8 v131(p2,p2,out131);
booth v132(out131,out131,out132);
booth8 v133(q2,q2,out133);
booth v134(out133,out133,out134);
booth8 v135(r2,r2,out135);
booth v136(out135,out135,out136);
booth8 v137(s2,s2,out137);
booth v138(out137,out137,out138);
booth8 v139(t2,t2,out139);
booth v140(out139,out139,out140);
booth8 v141(u2,u2,out141);
booth v142(out141,out141,out142);
booth8 v143(v2,v2,out143);
booth v144(out143,out143,out144);
booth8 v145(w2,w2,out145);
booth v146(out145,out145,out146);
booth8 v147(y2,y2,out147);
booth v148(out147,out147,out148);
booth8 v149(z2,z2,out149);
booth v150(out149,out149,out150);
booth8 v151(a3,a3,out151);
booth v152(out151,out151,out152);
booth8 v153(b3,b3,out153);
```

```
booth v154(out153,out153,out154);
booth8 v155(c3,c3,out155);
booth v156(out155,out155,out156);
booth8 v157(d3,d3,out157);
booth v158(out157,out157,out158);
booth8 v159(e3,e3,out159);
booth v160(out159,out159,out160);
booth8 v161(f3,f3,out161);
booth v162(out161,out161,out162);
booth8 v163(g3,g3,out163);
booth v164(out163,out163,out164);
booth8 v165(h3,h3,out165);
booth v166(out165,out165,out166);
booth8 v167(i3,i3,out167);
booth v168(out167,out167,out168);
booth8 v169(j3,j3,out169);
booth v170(out169,out169,out170);
booth8 v171(k3,k3,out171);
booth v172(out171,out171,out172);
booth8 v173(l3,l3,out173);
booth v174(out173,out173,out174);
booth8 v175(m3,m3,out175);
booth v176(out175,out175,out176);
booth8 v177(n3,n3,out177);
booth v178(out177,out177,out178);
booth8 v179(o3,o3,out179);
booth v180(out179,out179,out180);
booth8 v181(p3,p3,out181);
```

```
booth v182(out181,out181,out182);
booth8 v183(q3,q3,out183);
booth v184(out183,out183,out184);
booth8 v185(r3,r3,out185);
booth v186(out185,out185,out186);
booth8 v187(s3,s3,out187);
booth v188(out187,out187,out188);
booth8 v189(t3,t3,out189);
booth v190(out189,out189,out190);
booth8 v191(u3,u3,out191);
booth v192(out191,out191,out192);
booth8 v193(v3,v3,out193);
booth v194(out193,out193,out194);
booth8 v195(w3,w3,out195);
booth v196(out195,out195,out196);
booth8 v197(y3,y3,out197);
booth v198(out197,out197,out198);
booth8 v199(z3,z3,out199);
booth v200(out199,out199,out200);
booth8 v201(a4,a4,out201);
booth v202(out201,out201,out202);
booth8 v203(b4,b4,out203);
booth v204(out203,out203,out204);
booth8 v205(c4,c4,out205);
booth v206(out205,out205,out206);
booth8 v207(d4,d4,out207);
booth v208(out207,out207,out208);
booth8 v209(e4,e4,out209);
```

```
booth v210(out209,out209,out210);
booth8 v211(f4,f4,out211);
booth v212(out211,out211,out212);
booth8 v213(g4,g4,out213);
booth v214(out213,out213,out214);
booth8 v215(h4,h4,out215);
booth v216(out215,out215,out216);
booth8 v217(i4,i4,out217);
booth v218(out217,out217,out218);
booth8 v219(j4,j4,out219);
booth v220(out219,out219,out220);
booth8 v221(k4,k4,out221);
booth v222(out221,out221,out222);
booth8 v223(l4,l4,out223);
booth v224(out223,out223,out224);
booth8 v225(m4,m4,out225);
booth v226(out225,out225,out226);
booth8 v227(n4,n4,out227);
booth v228(out227,out227,out228);
booth8 v229(o4,o4,out229);
booth v230(out229,out229,out230);
booth8 v231(p4,p4,out231);
booth v232(out231,out231,out232);
booth8 v233(q4,q4,out233);
booth v234(out233,out233,out234);
booth8 v235(r4,r4,out235);
booth v236(out235,out235,out236);
booth8 v237(s4,s4,out237);
```

```
booth v238(out237,out237,out238);
booth8 v239(t4,t4,out239);
booth v240(out239,out239,out240);
booth8 v241(u4,u4,out241);
booth v242(out241,out241,out242);
booth8 v243(v4,v4,out243);
booth v244(out243,out243,out244);
booth8 v245(w4,w4,out245);
booth v246(out245,out245,out246);
booth8 v247(y4,y4,out247);
booth v248(out247,out247,out248);
booth8 v249(z4,z4,out249);
booth v250(out249,out249,out250);
booth8 v251(a5,a5,out251);
booth v252(out251,out251,out252);
booth8 v253(b5,b5,out253);
booth v254(out253,out253,out254);
booth8 v255(c5,c5,out255);
booth v256(out255,out255,out256);
booth8 v257(d5,d5,out257);
booth v258(out257,out257,out258);
booth8 v259(e5,e5,out259);
booth v260(out259,out259,out260);
booth8 v261(f5,f5,out261);
booth v262(out261,out261,out262);
booth8 v263(g5,g5,out263);
booth v264(out263,out263,out264);
booth8 v265(h5,h5,out265);
```

```
booth v266(out265,out265,out266);
booth8 v267(i5,i5,out267);
booth v268(out267,out267,out268);
booth8 v269(j5,j5,out269);
booth v270(out269,out269,out270);
booth8 v271(k5,k5,out271);
booth v272(out271,out271,out272);
booth8 v273(l5,l5,out273);
booth v274(out273,out273,out274);
booth8 v275(m5,m5,out275);
booth v276(out275,out275,out276);
booth8 v277(n5,n5,out277);
booth v278(out277,out277,out278);
booth8 v279(o5,o5,out279);
booth v280(out279,out279,out280);
booth8 v281(p5,p5,out281);
booth v282(out281,out281,out282);
booth8 v283(q5,q5,out283);
booth v284(out283,out283,out284);
booth8 v285(r5,r5,out285);
booth v286(out285,out285,out286);
booth8 v287(s5,s5,out287);
booth v288(out287,out287,out288);
booth8 v289(t5,t5,out289);
booth v290(out289,out289,out290);
booth8 v291(u5,u5,out291);
booth v292(out291,out291,out292);
booth8 v293(v5,v5,out293);
```

```
booth v294(out293,out293,out294);
booth8 v295(w5,w5,out295);
booth v296(out295,out295,out296);
booth8 v297(y5,y5,out297);
booth v298(out297,out297,out298);
booth8 v299(z5,z5,out299);
booth v300(out299,out299,out300);
booth8 v301(a6,a6,out301);
booth v302(out301,out301,out302);
booth8 v303(b6,b6,out303);
booth v304(out303,out303,out304);
booth8 v305(c6,c6,out305);
booth v306(out305,out305,out306);
booth8 v307(d6,d6,out307);
booth v308(out307,out307,out308);
booth8 v309(e6,e6,out309);
booth v310(out309,out309,out310);
booth8 v311(f6,f6,out311);
booth v312(out311,out311,out312);
booth8 v313(g6,g6,out313);
booth v314(out313,out313,out314);
booth8 v315(h6,h6,out315);
booth v316(out315,out315,out316);
booth8 v317(i6,i6,out317);
booth v318(out317,out317,out318);
booth8 v319(j6,j6,out319);
booth v320(out319,out319,out320);
booth8 v321(k6,k6,out321);
```

```
booth v322(out321,out321,out322);
booth8 v323(l6,l6,out323);
booth v324(out323,out323,out324);
booth8 v325(m6,m6,out325);
booth v326(out325,out325,out326);
booth8 v327(n6,n6,out327);
booth v328(out327,out327,out328);
booth8 v329(o6,o6,out329);
booth v330(out329,out329,out330);
booth8 v331(p6,p6,out331);
booth v332(out331,out331,out332);
booth8 v333(q6,q6,out333);
booth v334(out333,out333,out334);
booth8 v335(r6,r6,out335);
booth v336(out335,out335,out336);
booth8 v337(s6,s6,out337);
booth v338(out337,out337,out338);
booth8 v339(t6,t6,out339);
booth v340(out339,out339,out340);
booth8 v341(u6,u6,out341);
booth v342(out341,out341,out342);
booth8 v343(v6,v6,out343);
booth v344(out343,out343,out344);
booth8 v345(w6,w6,out345);
booth v346(out345,out345,out346);
booth8 v347(y6,y6,out347);
booth v348(out347,out347,out348);
booth8 v349(z6,z6,out349);
```

```
booth v350(out349,out349,out350);
booth8 v351(a7,a7,out351);
booth v352(out351,out351,out352);
booth8 v353(b7,b7,out353);
booth v354(out353,out353,out354);
booth8 v355(c7,c7,out355);
booth v356(out355,out355,out356);
booth8 v357(d7,d7,out357);
booth v358(out357,out357,out358);
booth8 v359(e7,e7,out359);
booth v360(out359,out359,out360);
booth8 v361(f7,f7,out361);
booth v362(out361,out361,out362);
booth8 v363(g7,g7,out363);
booth v364(out363,out363,out364);
booth8 v365(h7,h7,out365);
booth v366(out365,out365,out366);
booth8 v367(i7,i7,out367);
booth v368(out367,out367,out368);
booth8 v369(j7,j7,out369);
booth v370(out369,out369,out370);
booth8 v371(k7,k7,out371);
booth v372(out371,out371,out372);
booth8 v373(l7,l7,out373);
booth v374(out373,out373,out374);
booth8 v375(m7,m7,out375);
booth v376(out375,out375,out376);
booth8 v377(n7,n7,out377);
```

```
booth v378(out377,out377,out378);
booth8 v379(o7,o7,out379);
booth v380(out379,out379,out380);
booth8 v381(p7,p7,out381);
booth v382(out381,out381,out382);
booth8 v383(q7,q7,out383);
booth v384(out383,out383,out384);
booth8 v385(r7,r7,out385);
booth v386(out385,out385,out386);
booth8 v387(s7,s7,out387);
booth v388(out387,out387,out388);
booth8 v389(t7,t7,out389);
booth v390(out389,out389,out390);
booth8 v391(u7,u7,out391);
booth v392(out391,out391,out392);
booth8 v393(v7,v7,out393);
booth v394(out393,out393,out394);
booth8 v395(w7,w7,out395);
booth v396(out395,out395,out396);
booth8 v397(y7,y7,out397);
booth v398(out397,out397,out398);
booth8 v399(z7,z7,out399);
booth v400(out399,out399,out400);
booth8 v401(a8,a8,out401);
booth v402(out401,out401,out402);
booth8 v403(b8,b8,out403);
booth v404(out403,out403,out404);
booth8 v405(c8,c8,out405);
```

```
booth v406(out405,out405,out406);
booth8 v407(d8,d8,out407);
booth v408(out407,out407,out408);
booth8 v409(e8,e8,out409);
booth v410(out409,out409,out410);
booth8 v411(f8,f8,out411);
booth v412(out411,out411,out412);
booth8 v413(g8,g8,out413);
booth v414(out413,out413,out414);
booth8 v415(h8,h8,out415);
booth v416(out415,out415,out416);
booth8 v417(i8,i8,out417);
booth v418(out417,out417,out418);
booth8 v419(j8,j8,out419);
booth v420(out419,out419,out420);
booth8 v421(k8,k8,out421);
booth v422(out421,out421,out422);
booth8 v423(l8,l8,out423);
booth v424(out423,out423,out424);
booth8 v425(m8,m8,out425);
booth v426(out425,out425,out426);
booth8 v427(n8,n8,out427);
booth v428(out427,out427,out428);
booth8 v429(o8,o8,out429);
booth v430(out429,out429,out430);
booth8 v431(p8,p8,out431);
booth v432(out431,out431,out432);
booth8 v433(q8,q8,out433);
```

```
booth v434(out433,out433,out434);
booth8 v435(r8,r8,out435);
booth v436(out435,out435,out436);
booth8 v437(s8,s8,out437);
booth v438(out437,out437,out438);
booth8 v439(t8,t8,out439);
booth v440(out439,out439,out440);
booth8 v441(u8,u8,out441);
booth v442(out441,out441,out442);
booth8 v443(v8,v8,out443);
booth v444(out443,out443,out444);
booth8 v445(w8,w8,out445);
booth v446(out445,out445,out446);
booth8 v447(y8,y8,out447);
booth v448(out447,out447,out448);
booth8 v449(z8,z8,out449);
booth v450(out449,out449,out450);
booth8 v451(a9,a9,out451);
booth v452(out451,out451,out452);
booth8 v453(b9,b9,out453);
booth v454(out453,out453,out454);
booth8 v455(c9,c9,out455);
booth v456(out455,out455,out456);
booth8 v457(d9,d9,out457);
booth v458(out457,out457,out458);
booth8 v459(e9,e9,out459);
booth v460(out459,out459,out460);
booth8 v461(f9,f9,out461);
```

```
booth v462(out461,out461,out462);
booth8 v463(g9,g9,out463);
booth v464(out463,out463,out464);
booth8 v465(h9,h9,out465);
booth v466(out465,out465,out466);
booth8 v467(i9,i9,out467);
booth v468(out467,out467,out468);
booth8 v469(j9,j9,out469);
booth v470(out469,out469,out470);
booth8 v471(k9,k9,out471);
booth v472(out471,out471,out472);
booth8 v473(l9,l9,out473);
booth v474(out473,out473,out474);
booth8 v475(m9,m9,out475);
booth v476(out475,out475,out476);
booth8 v477(n9,n9,out477);
booth v478(out477,out477,out478);
booth8 v479(o9,o9,out479);
booth v480(out479,out479,out480);
booth8 v481(p9,p9,out481);
booth v482(out481,out481,out482);
booth8 v483(q9,q9,out483);
booth v484(out483,out483,out484);
booth8 v485(r9,r9,out485);
booth v486(out485,out485,out486);
booth8 v487(s9,s9,out487);
booth v488(out487,out487,out488);
booth8 v489(t9,t9,out489);
```

```
booth v490(out489,out489,out490);
booth8 v491(u9,u9,out491);
booth v492(out491,out491,out492);
booth8 v493(v9,v9,out493);
booth v494(out493,out493,out494);
booth8 v495(w9,w9,out495);
booth v496(out495,out495,out496);
booth8 v497(y9,y9,out497);
booth v498(out497,out497,out498);
booth8 v499(z9,z9,out499);
booth v500(out499,out499,out500);
booth8 v501(a10,a10,out501);
booth v502(out501,out501,out502);
booth8 v503(b10,b10,out503);
booth v504(out503,out503,out504);
booth8 v505(c10,c10,out505);
booth v506(out505,out505,out506);
booth8 v507(d10,d10,out507);
booth v508(out507,out507,out508);
booth8 v509(e10,e10,out509);
booth v510(out509,out509,out510);
booth8 v511(f10,f10,out511);
booth v512(out511,out511,out512);
add v513(out2,out4,out6,out8,out10,out12,out14,out16,aa);
add v514(out18,out20,out22,out24,out26,out28,out30,out32,bb);
add v515(out34,out36,out38,out40,out42,out44,out46,out48,cc);
add v516(out50,out52,out54,out56,out58,out60,out62,out64,dd);
add v517(out66,out68,out70,out72,out74,out76,out78,out80,ee);
```

```
add v518(out82,out84,out86,out88,out90,out92,out94,out96,ff);
add v519(out98,out100,out102,out104,out106,out108,out110,out112,gg);
add v520(out114,out116,out118,out120,out122,out124,out126,out128,hh);
add v521(out130,out132,out134,out136,out138,out140,out142,out144,ii);
add v522(out146,out148,out150,out152,out154,out156,out158,out160,jj);
add v523(out162,out164,out166,out168,out170,out172,out174,out176,kk);
add v524(out178,out180,out182,out184,out186,out188,out190,out192,ll);
add v525(out194,out196,out198,out200,out202,out204,out206,out208,mmm);
add v526(out210,out212,out214,out216,out218,out220,out222,out224,nn);
add v527(out226,out228,out230,out232,out234,out236,out238,out240,oo);
add v528(out242,out244,out246,out248,out250,out252,out254,out256,pp);
add v529(out258,out260,out262,out264,out266,out268,out270,out272,qq);
add v530(out274,out276,out278,out280,out282,out284,out286,out288,rr);
add v531(out290,out292,out294,out296,out298,out300,out302,out304,ss);
add v532(out306,out308,out310,out312,out314,out316,out318,out320,tt);
add v533(out322,out324,out326,out328,out330,out332,out334,out336,uu);
add v534(out338,out340,out342,out344,out346,out348,out350,out352,vv);
add v535(out354,out356,out358,out360,out362,out364,out366,out368,ww);
add v536(out370,out372,out374,out376,out378,out380,out382,out384,xx);
add v537(out386,out388,out390,out392,out394,out396,out398,out400,yy);
add v538(out402,out404,out406,out408,out410,out412,out414,out416,zz);
add v539(out418,out420,out422,out424,out426,out428,out430,out432,aa1);
add v540(out434,out436,out438,out440,out442,out444,out446,out448,bb1);
add v541(out450,out452,out454,out456,out458,out460,out462,out464,cc1);
add v542(out466,out468,out470,out472,out474,out476,out478,out480,dd1);
add v543(out482,out484,out486,out488,out490,out492,out494,out496,ee1);
add v544(out498,out500,out502,out504,out506,out508,out510,out512,ff1);
add v545(aa,bb,cc,dd,ee,ff,gg,hh,res1);
```

```
 add v546(ii,jj,kk,ll,mmm,nn,oo,pp,res2);
 add v547(qq,rr,ss,tt,uu,vv,ww,xx,res3);
 add v548(yy,zz,aa1,bb1,cc1,dd1,ee1,ff1,res4);
 ripple v549(res1,res2,ra,cin1,res5);
 ripple v550(res3,res4,ra,cin2,res6);
 ripple v551(res5,res6,ra,cin3,res7);
 var v552(clk,out513);
 booth v553(out513,out513,out514);
 assign out=res7/out514;
 endmodule
module text1(clk,u);
  input clk;
 output [7:0]u;
  reg [11:0] address;
  reg [7:0] m[255:0];
  wire[7:0]avg;
  wire[7:0]avg1;
  wire[7:0]avg2;
  wire[7:0]avg3;
  wire[7:0]avg4;
  wire[7:0]avg5;
  wire[7:0]avg6;
  wire[7:0]avg7;
  wire[7:0]avg8;
  wire[7:0]avg9;
  wire[7:0]avg10;
  wire[7:0]avg11;
  wire[7:0]avg12;
```

```
 wire[7:0]avg13;
 wire[7:0]avg14;
 wire[7:0]avg15;
 wire[7:0]avg16;
 wire[7:0]avg17;
 wire[7:0]avg18;
 wire[7:0]avg19;
 wire[7:0]avg20;
 wire[7:0]avg21;
 wire[7:0]avg22;
 wire[7:0]avg23;
 wire[7:0]avg24;
 wire[7:0]avg25;
 wire[7:0]avg26;
 wire[7:0]avg27;
 wire[7:0]avg28;
 wire[7:0]avg29;
 wire[7:0]avg30;
 wire[7:0]avg31;
 wire[7:0]p;
 wire[7:0]q;
 wire[7:0]r;
 wire[7:0]s;
 integer i=0;
 initial
 begin
 $readmemb("D:/rashmi pjt/rr.txt",m);
end
```

```
initial address=12'b000000000000;
always@(posedge clk)
begin
  address<=address+1'b1;
end
refaverage8 u1(avg, m[0],m[1],m[2],m[3],m[4],m[5],m[6],m[7]);
refaverage8 u2(avg1, m[8],m[9],m[10],m[11],m[12],m[13],m[14],m[15]);
refaverage8 u3(avg2, m[16],m[17],m[18],m[19],m[20],m[21],m[22],m[23]);
refaverage8 u4(avg3, m[24],m[25],m[26],m[27],m[28],m[29],m[30],m[31]);
refaverage8 u5(avg4, m[32],m[33],m[34],m[35],m[36],m[37],m[38],m[39]);
refaverage8 u6(avg5, m[40],m[41],m[42],m[43],m[44],m[45],m[46],m[47]);
refaverage8 u7(avg6, m[48],m[49],m[50],m[51],m[52],m[53],m[54],m[55]);
refaverage8 u8(avg7, m[56],m[57],m[58],m[59],m[60],m[61],m[62],m[63]);
refaverage8 u9(avg8, m[64],m[65],m[66],m[67],m[68],m[69],m[70],m[71]);
refaverage8 u10(avg9, m[72],m[73],m[74],m[75],m[76],m[77],m[78],m[79]);
refaverage8 u11(avg10, m[80],m[81],m[82],m[83],m[84],m[85],m[86],m[87]);
refaverage8 u12(avg11, m[88],m[89],m[90],m[91],m[92],m[93],m[94],m[95]);
refaverage8 u13(avg12, m[96],m[97],m[98],m[99],m[100],m[101],m[102],m[103]);
refaverage8 u14(avg13, m[104],m[105],m[106],m[107],m[108],m[109],m[110],m[111]);
refaverage8 u15(avg14, m[112],m[113],m[114],m[115],m[116],m[117],m[118],m[119]);
refaverage8 u16(avg15, m[120],m[121],m[122],m[123],m[124],m[125],m[126],m[127]);
refaverage8 u17(avg16, m[128],m[129],m[130],m[131],m[132],m[133],m[134],m[135]);
refaverage8 u18(avg17, m[136],m[137],m[138],m[139],m[140],m[141],m[142],m[143]);
refaverage8 u19(avg18, m[144],m[145],m[146],m[147],m[148],m[149],m[150],m[151]);
refaverage8 u20(avg19, m[152],m[153],m[154],m[155],m[156],m[157],m[158],m[159]);
refaverage8 u21(avg20, m[160],m[161],m[162],m[163],m[164],m[165],m[166],m[167]);
refaverage8 u22(avg21, m[168],m[169],m[170],m[171],m[172],m[173],m[174],m[175]);
refaverage8 u23(avg22, m[176],m[177],m[178],m[179],m[180],m[181],m[182],m[183]);
```

```
  refaverage8 u24(avg23, m[184],m[185],m[186],m[187],m[188],m[189],m[190],m[191]);
  refaverage8 u25(avg24, m[192],m[193],m[194],m[195],m[196],m[197],m[198],m[199]);
  refaverage8 u26(avg25, m[200],m[201],m[202],m[203],m[204],m[205],m[206],m[207]);
  refaverage8 u27(avg26, m[208],m[209],m[210],m[211],m[212],m[213],m[214],m[215]);
  refaverage8 u28(avg27, m[216],m[217],m[218],m[219],m[220],m[221],m[222],m[223]);
  refaverage8 u29(avg28, m[224],m[225],m[226],m[227],m[228],m[229],m[230],m[231]);
  refaverage8 u30(avg29, m[232],m[233],m[234],m[235],m[236],m[237],m[238],m[239]);
  refaverage8 u31(avg30, m[240],m[241],m[242],m[243],m[244],m[245],m[246],m[247]);
  refaverage8 u32(avg31, m[248],m[249],m[250],m[251],m[252],m[253],m[254],m[255]);
  refaverage8 u34(p,avg,avg1,avg2,avg3,avg4,avg5,avg6,avg7);
  refaverage8 u35(q,avg8,avg9,avg10,avg11,avg12,avg13,avg14,avg15);
  refaverage8 u36(r,avg16,avg17,avg18,avg19,avg20,avg21,avg22,avg23);
  refaverage8 u37(s,avg24,avg25,avg26,avg27,avg28,avg29,avg30,avg31);
  refaverage4 u38(u,p,q,r,s);
endmodule
%% Ripple counter
module ripple(a,b,cin,carry,sum);
input [7:0]a;
input [7:0]b;
input cin;
output carry;
output [7:0]sum;
wire c1,c2,c3,c4,c5,c6,c7;
fa d(a[0],b[0],cin,c1,sum[0]);
fa d1(a[1],b[1],c1,c2,sum[1]);
fa d2(a[2],b[2],c2,c3,sum[2]);
fa d3(a[3],b[3],c3,c4,sum[3]);
fa d4(a[4],b[4],c4,c5,sum[4]);
```

```
fa d5(a[5],b[5],c5,c6,sum[5]);

fa d6(a[6],b[6],c6,c7,sum[6]);

fa d7(a[7],b[7],c7,carry,sum[7]);

endmodule

module fa(a,b,cin,carry,sum);

input a,b,cin;

output carry,sum;

assign carry=((a&b)|(b&cin)|(a&cin));

assign sum=a^b^cin;

endmodule

module ripple(a,b,cin,carry,sum);

input [31:0]a;

input [31:0]b;

input cin;

output carry;

output [31:0]sum;

wire
c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,c13,c14,c15,c16,c17,c18,c19,c20,c21,c22,c23,c24,c25,c26,c27,c28
,c29,c30,c31;

fa d(a[0],b[0],cin,c1,sum[0]);

fa d1(a[1],b[1],c1,c2,sum[1]);

fa d2(a[2],b[2],c2,c3,sum[2]);

fa d3(a[3],b[3],c3,c4,sum[3]);

fa d4(a[4],b[4],c4,c5,sum[4]);

fa d5(a[5],b[5],c5,c6,sum[5]);

fa d6(a[6],b[6],c6,c7,sum[6]);

fa d7(a[7],b[7],c7,c8,sum[7]);

fa d8(a[8],b[8],c8,c9,sum[8]);
```

```
fa d9(a[9],b[9],c9,c10,sum[9]);
fa d10(a[10],b[10],c10,c11,sum[10]);
fa d11(a[11],b[11],c11,c12,sum[11]);
fa d12(a[12],b[12],c12,c13,sum[12]);
fa d13(a[13],b[13],c13,c14,sum[13]);
fa d14(a[14],b[14],c14,c15,sum[14]);
fa d15(a[15],b[15],c15,c16,sum[15]);
fa d16(a[16],b[16],c16,c17,sum[16]);
fa d17(a[17],b[17],c17,c18,sum[17]);
fa d18(a[18],b[18],c18,c19,sum[18]);
fa d19(a[19],b[19],c19,c20,sum[19]);
fa d20(a[20],b[20],c20,c21,sum[20]);
fa d21(a[21],b[21],c21,c22,sum[21]);
fa d22(a[22],b[22],c22,c23,sum[22]);
fa d23(a[23],b[23],c23,c24,sum[23]);
fa d24(a[24],b[24],c24,c25,sum[24]);
fa d25(a[25],b[25],c25,c26,sum[25]);
fa d26(a[26],b[26],c26,c27,sum[26]);
fa d27(a[27],b[27],c27,c28,sum[27]);
fa d28(a[28],b[28],c28,c29,sum[28]);
fa d29(a[29],b[29],c29,c30,sum[29]);
fa d30(a[30],b[30],c30,c31,sum[30]);
fa d31(a[31],b[31],c31,carry,sum[31]);
endmodule
%% Test
    module test1(clk,x1,x2,x3,out);
 input clk;
 input [9:0]x1,x2,x3;
```

```
output reg[1:0]out;
wire[1:0]out1,out2,out3;
integer count1=0;
integer count2=0;
train u1(clk,x1,out1);
train u2(clk,x2,out2);
train u3(clk,x3,out3);
always@(posedge clk)
  begin
if((out1==out2)||(out1==out3))
  count1=count1+1;
else if(out2==out3)
  count2=count2+1;
end
always@(posedge clk)
  begin
if(count1>count2)
  begin
  out=out1;
  //count1=0;
  //count2=0;
end
else
  begin
  out=out2;
  //count1=0;
  //count2=0;
end
```

```
  end
endmodule
 %% Train
  module train(clk,in,out);
  input clk;
  wire clka;
  input[9:0]in;
  wire wea;
  output reg [1:0]out;
 //reg [1:0]out1;
wire [2 : 0] addra;
wire [9 : 0] dina;
wire [9 : 0] douta;
/*reg [9:0]m[5:0];
reg [9:0]n[5:0];
reg [9:0]o[5:0];
reg [9:0]p[5:0];
reg [9:0]q[5:0];
reg [9:0]r[5:0];
reg [9:0]s[5:0];
reg [9:0]t[5:0];
reg [9:0]u[5:0];*/
m u10(clka,rsta,wea,addra,dina,douta);
n u2(clka,rsta,wea,addra,dina,douta);
o u3(clka,rsta,wea,addra,dina,douta);
p u4(clka,rsta,wea,addra,dina,douta);
q u5(clka,rsta,wea,addra,dina,douta);
r u6(clka,rsta,wea,addra,dina,douta);
```

```
s u7(clka,rsta,wea,addra,dina,douta);

t u8(clka,rsta,wea,addra,dina,douta);

u u9(clka,rsta,wea,addra,dina,douta);

/*always@(posedge clk)

begin

$readmemb("D:/rohini/rash pjt/mean norm.txt",m);

$readmemb("D:/rohini/rash pjt/var norm.txt",n);

$readmemb("D:/rohini/rash pjt/kur norm.txt",o);

$readmemb("D:/rohini/rash pjt/mean cyst.txt",p);

$readmemb("D:/rohini/rash pjt/var cyst.txt",q);

$readmemb("D:/rohini/rash pjt/kur cyst.txt",r);

$readmemb("D:/rohini/rash pjt/mean stone.txt",s);

$readmemb("D:/rohini/rash pjt/var stone.txt",t);

$readmemb("D:/rohini/rash pjt/kur stone.txt",u);

end*/

 /*always@(posedge clk)

begin

  if((in==douta1[0])|(in==douta1[1])|(in==douta1[2])|(in==douta1[3])|(in==douta1[4])|(in==douta1[5]))

    out=2'b00;

  else
if((in==douta2[0])|(in==douta2[1])|(in==douta2[2])|(in==douta2[3])|(in==douta2[4])|(in==douta2[5]))

    out=2'b00;

  else
if((in==douta3[0])|(in==douta3[1])|(in==douta3[2])|(in==douta3[3])|(in==douta3[4])|(in==douta3[5]))

    out=2'b00;

    else
if((in==douta4[0])|(in==douta4[1])|(in==douta4[2])|(in==douta4[3])|(in==douta4[4])|(in==douta4[5]))

    out=2'b01;
```

```
  else
if((in==douta5[0])|(in==douta5[1])|(in==douta5[2])|(in==douta5[3])|(in==douta5[4])|(in==douta5[5]))

    out=2'b01;

  else
if((in==douta6[0])|(in==douta6[1])|(in==douta6[2])|(in==douta6[3])|(in==douta6[4])|(in==douta6[5]))

    out=2'b01;

  else
if((in==douta7[0])|(in==douta7[1])|(in==douta7[2])|(in==douta7[3])|(in==douta7[4])|(in==douta7[5]))

    out=2'b10;

  else
if((in==douta8[0])|(in==douta8[1])|(in==douta8[2])|(in==douta8[3])|(in==douta8[4])|(in==douta8[5]))

    out=2'b10;

  else
if((in==douta9[0])|(in==douta9[1])|(in==douta9[2])|(in==douta9[3])|(in==douta9[4])|(in==douta9[5]))

    out=2'b10;

            else

            out=2'b00;

    end*/

/*always@(posedge clk)

begin

  if((in==m[0])|(in==m[1])|(in==m[2])|(in==m[3])|(in==m[4])|(in==m[5]))

    out=2'b00;

  else if((in==n[0])|(in==n[1])|(in==n[2])|(in==n[3])|(in==n[4])|(in==n[5]))

    out=2'b00;

  else if((in==o[0])|(in==o[1])|(in==o[2])|(in==o[3])|(in==o[4])|(in==o[5]))

    out=2'b00;

    else if((in==p[0])|(in==p[1])|(in==p[2])|(in==p[3])|(in==p[4])|(in==p[5]))

    out=2'b01;

  else if((in==q[0])|(in==q[1])|(in==q[2])|(in==q[3])|(in==q[4])|(in==q[5]))

    out=2'b01;
```

```
else if((in==r[0])|(in==r[1])|(in==r[2])|(in==r[3])|(in==r[4])|(in==r[5]))
  out=2'b01;
else if((in==s[0])|(in==s[1])|(in==s[2])|(in==s[3])|(in==s[4])|(in==s[5]))
  out=2'b10;
else if((in==t[0])|(in==t[1])|(in==t[2])|(in==t[3])|(in==t[4])|(in==t[5]))
  out=2'b10;
else if((in==u[0])|(in==u[1])|(in==u[2])|(in==u[3])|(in==u[4])|(in==u[5]))
  out=2'b10;
//else
  //out=2'b11;
  end*/
  always@(posedge clk)
  begin
   if(in<10'b0000000010)
     out=2'b10;
   else if((10'b0000000010<=in)&&(in<=10'b0000000100))
     out=2'b00;
   else if((10'b0000000101<=in)&&(in<=10'b0000001010))
     out=2'b01;
   else if((10'b0000001011<=in)&&(in<=10'b0000100111))
     out=2'b10;
   else if((10'b0000101000<=in)&&(in<=10'b0001000110))
     out=2'b00;
   else if((10'b0010000110<=in)&&(in<=10'b0101000000))
     out=2'b00;
   else if((10'b0101000001<=in)&&(in<=10'b1100110010))
     out=2'b01;
   // else if(10'd11<=in<=10'd39)
```

```
      //out1=2'b01;
    else if((10'b0001000111<=in)&&(in<=10'b0010000110)||(in>=10'b1100110011))
      out=2'b10;
    //else if(in>=10'd819)
      //out=2'b10;
    //else if(in>=10'd71);
      //out1=2'b10;
    end
endmodule
```