Lab 8
Speech and Image Processing

Group Members:

Muhammad Rehman Rabbani	**137364**
Abdul Ghaffar Kalhoro	**194699**
Hamad Nasir	**120312**

Group assignment up to 3 students per group.
Using MATLAB or Octave, implement flood fill mechanism for colouring an image. For an example, see images 1 and 2 below.

Part (a) 2 marks
Create an RGB image of 400 x 600 pixels using Paint or Pinta (Linux), similar to image 1. Name it blob.bmp

Part (b) 2 marks
Read this image in MATLAB and convert it into an array of 600 x 400 x 3 double precision floating point numbers using double() function

Part (c) 6 marks
Implement flood fill mechanism, so that each blob in the original image is coloured with a random colour, like in image 2 below. Save it as blob2.bmp.

Hint:
This functionality cannot be achieved through 2 simple nested for loops. Try to think of some graph based, breadth first, or dynamic programming based algorithm, for example.
For colouring a blob: we start from one pixel, check its neighbours whether they have the same colour; if yes then colour them too and check the neighbours' neighbouring pixels and so on.

Deliverables:
- MATLAB code
- Images blob.bmp and blob2.bmp (these images should be unique for each group)

Code:

```
% reading original image
originalImage = imread('Image.png');
figure, imshow(originalImage);
title('Original Image');
% converting image to double
doubleImage = im2double(originalImage);
% converting doubled image to binary
binaryImage = im2bw(doubleImage);
% finding labels of binary image
% inverting binary image because background is white in input image
labelImage = BWLabel(~binaryImage);
% converting labled matrix to rgb image
rgb = label2rgb(labelImage, 'jet', 'm', 'shuffle');
figure, imshow(rgb);
title('RGB Image');
```

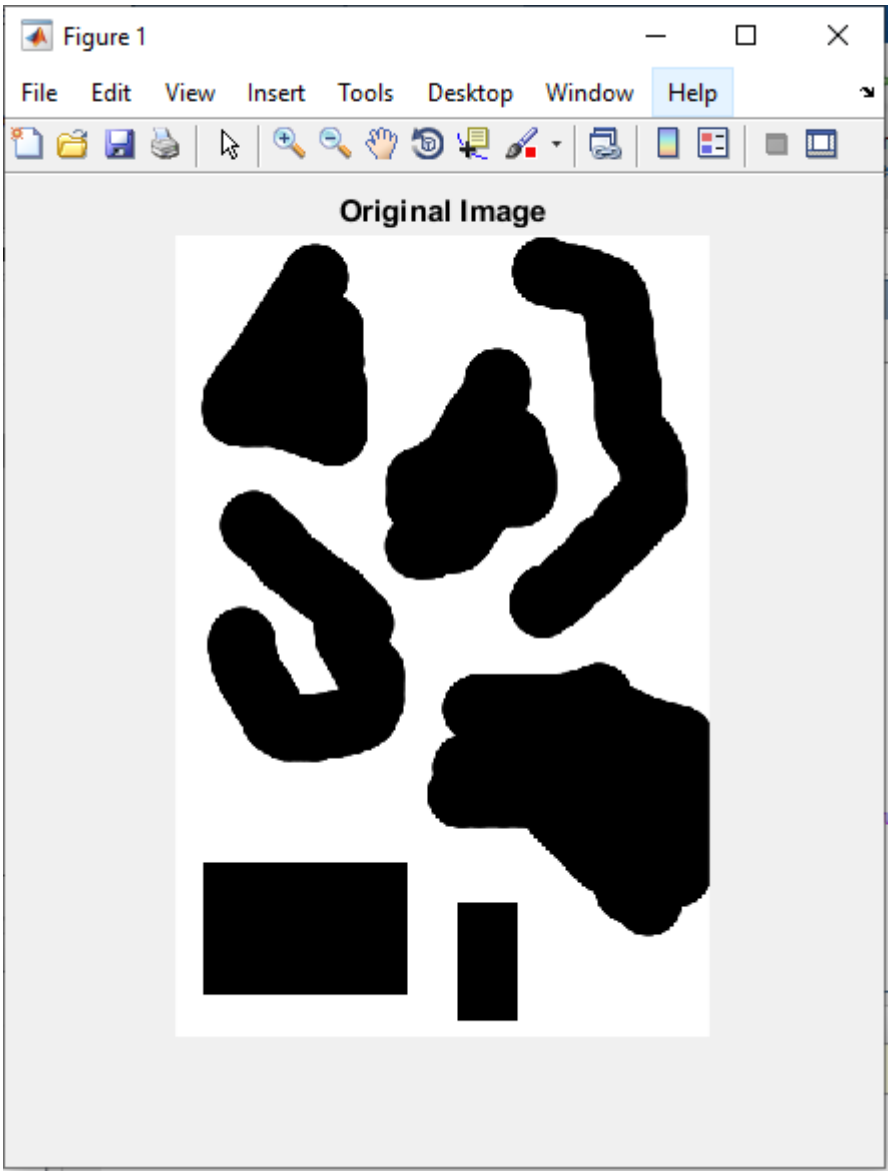
BWLabel.m:

```
% defining BWLabel function for binary image labeling
% Labels a binary image through 8-point connectivity
% this function is partly based on the two-pass algorithm
% it returns a labeled matrix
function [ labels ] = BWLabel( input )
% grtting dimensions of the input binary image
[x,y] = size(input);
% expanding the input set to avoid crash when searching
input = [zeros(1,y+2);[zeros(x,1) input zeros(x,1)]];
[x,y] = size(input);
% initializing label matrix with zeros
labels = zeros(size(input));

nextlabel = 1;
linked = [];
 % for each row
for i = 2:x
    % for each column
    for j = 2:y-1
        % not background
        if input(i,j) ~= 0
            % find binary value of neighbours (8-way connectivity)
            neighboursearch = [input(i-1,j-1), input(i-1,j), input(i-
1,j+1),input(i,j-1)];
            % search for neighbours with binary value 1
            [~,n,neighbours] = find(neighboursearch==1);

            % assign a new label, if no neighbour is already labeled
            if isempty(neighbours)
                linked{nextlabel} = nextlabel;
                labels(i,j) = nextlabel;
                nextlabel = nextlabel+1;

            % if neighbours is labeled: pick the lowest label and store the
            % connected labels in "linked"
            else
                neighboursearch_label = [labels(i-1,j-1), labels(i-1,j),
labels(i-1,j+1),labels(i,j-1)];
                L = neighboursearch_label(n);
                labels(i,j) = min(L);
                for k = 1:length(L)
                    label = L(k);
                    linked{label} = unique([linked{label} L]);
```

```
                end
            end
        end
    end
end
% remove the previous expansion of the image
labels = labels(2:end,2:end-1);

%% joining linked areas
% for each link, look through the other links and look for common labels.
% if common labels exist they are linked -> replace both link with the
% union of the two. Repeat until there is no change in the links.
change2 = 1;
while change2 == 1
    change = 0;
    for i = 1:length(linked)
        for j = 1:length(linked)
            if i ~= j
                if sum(ismember(linked{i},linked{j}))>0 &&
sum(ismember(linked{i},linked{j})) ~= length(linked{i})
                    change = 1;
                    linked{i} = union(linked{i},linked{j});
                    linked{j} = linked{i};
                end
            end
        end
    end

    if change == 0
        change2 = 0;
    end

end
% removing redundat links
linked = unique(cellfun(@num2str,linked,'UniformOutput',false));
linked = cellfun(@str2num,linked,'UniformOutput',false);
K = length(linked);
templabels = labels;
labels = zeros(size(labels));
% label linked labels with a single label:
for k = 1:K
    for l = 1:length(linked{k})
        labels(templabels == linked{k}(l)) = k;
    end
end
end
```

Output:

blob.bmp:



blob2.bmp:

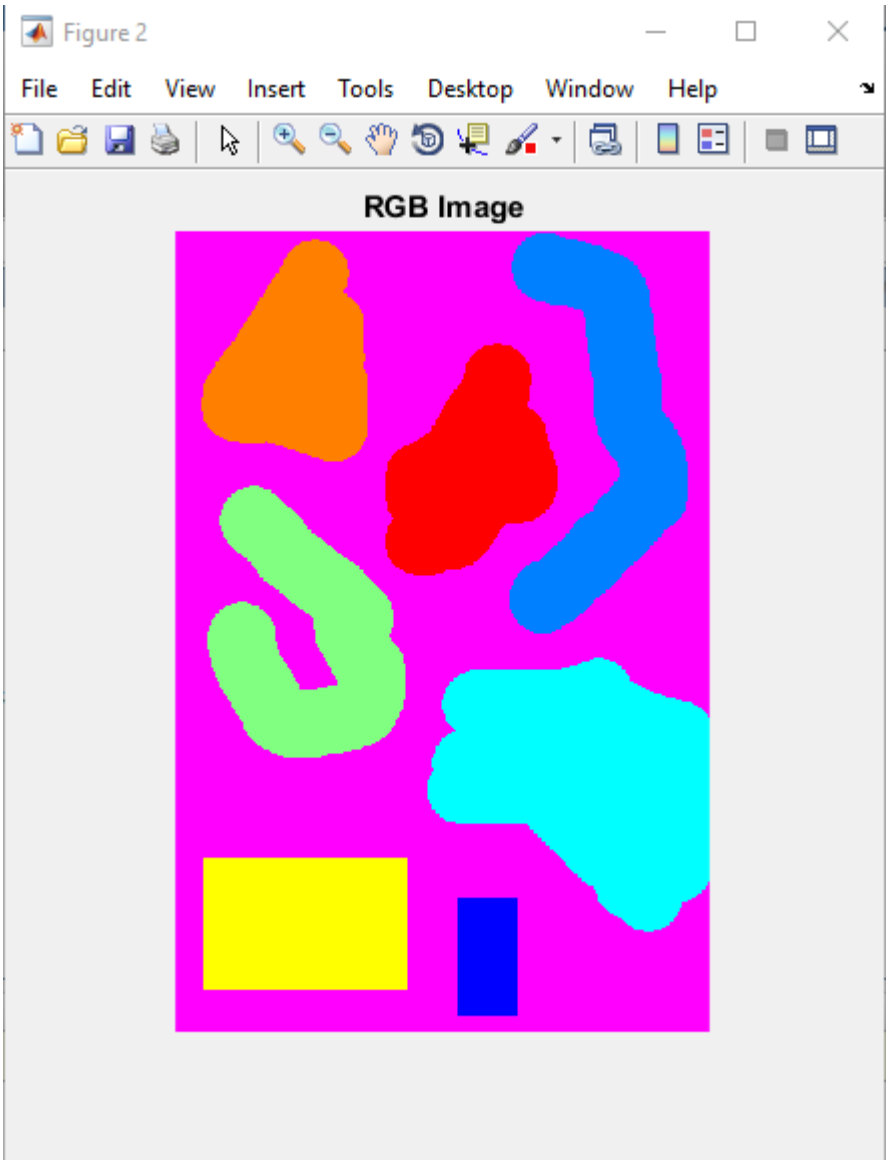


Figure 2
File Edit View Insert Tools Desktop Window Help
RGB Image