```
#include <vector>
#include <iostream>
#include <opencv2/opencv.hpp>

// Define a structure to represent a pixel in
the image
struct Pixel {
int row;
int col;

Pixel(int r, int c)   row(r), col(c) {}
};

// Function to perform a depth-first search
to find connected components
void DFS(const cv::Mat& binaryImage, int
row, int col, std::vector>& visited,
std::vector& component) {
    // Check boundaries and pixel value
if (row < 0          row >=
binaryImage.rows          col < 0
col >= binaryImage.cols          visited[row]
[col]          binaryImage.at(row, col) ==
0) {
return;
```

```
}

    Mark the pixel as visited
visited[row][col] = true;

    Add the pixel to the connected component
component.push_back(Pixel(row, col));

    Recursive calls to neighbors
DFS(binaryImage, row + 1, col, visited,
component);
DFS(binaryImage, row - 1, col, visited,
component);
DFS(binaryImage, row, col + 1, visited,
component);
DFS(binaryImage, row, col - 1, visited,
component);
}

// Function to find all connected
components in the binary image
std::vector>
findConnectedComponents(const cv::Mat&
binaryImage) {
int rows = binaryImage.rows;
```

```
int cols = binaryImage.cols;

std::vector> visited(rows, std::vector(cols,
false));
std::vector> components;

for (int row = 0; row < rows; ++row) {
for (int col = 0; col < cols; ++col) {
if (binaryImage.at(row, col) == 255 &
&visited[row][col]) {
    Start a new connected component
std::vector component;
DFS(binaryImage, row, col, visited,
component);
components.push_back(component);
            }
        }
    }

return components;
}

int main() {
    Load an image using OpenCV
cv::Mat inputImage =
```

```
cv::imread("/home/hi-born4/6th Sem/Image Processing and Computer Vision/16",
cv::IMREAD_GRAYSCALE);

if (inputImage.empty()) {
std::cerr << "Error: Unable to load the image!" << std::endl;
return -1;
    }

    Convert the image to binary (assuming it's a binary image with white foreground)
cv::Mat binaryImage;
cv::threshold(inputImage, binaryImage, 128, 255, cv::THRESH_BINARY);

    Find connected components in the binary image
std::vector> components =
FindConnectedComponents(binaryImage);

    Display the results
for (const auto& component  components)
{
std::cout << "Connected Component:" <<
```

```
std::endl;
for (const auto& pixel  component) {
std::cout << "(" << pixel.row << ", " <<
pixel.col << ") ";
        }
std::cout << std::endl;
    }


return 0;
}
```