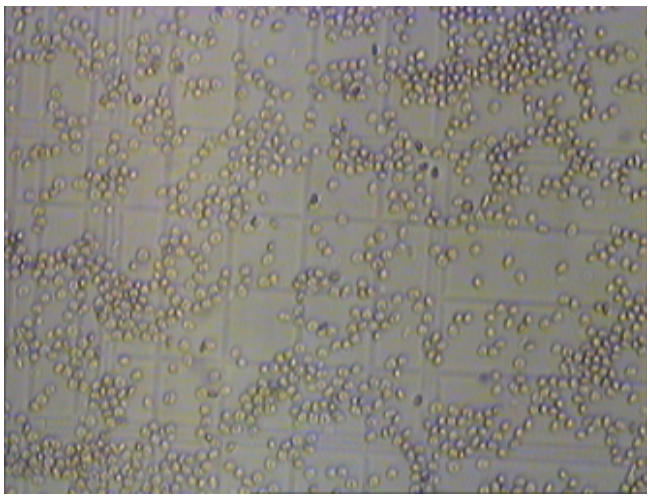
Assignment 1

Due Wednesday by 4pm **Points** 5

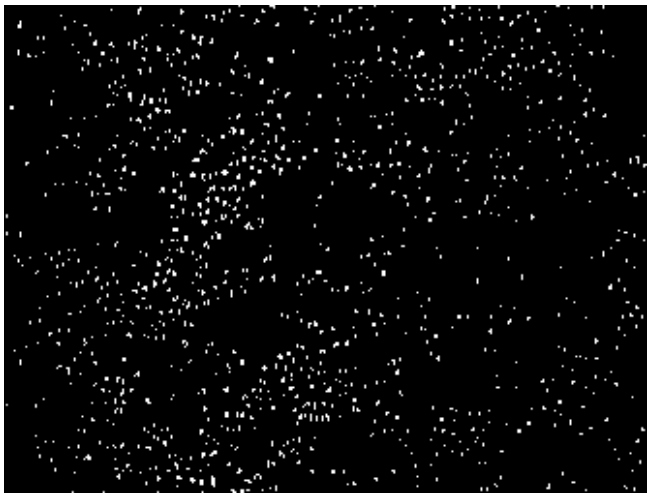
Introduction

Analyzing blood smear images, and extracting features of the cells is an important application in the field of medicine to help detect and monitor various types of diseases. In this assignment, you will implement a simple form of red blood cell counting. This assignment was inspired by an **article (https://www.sciencedirect.com/science/article/pii/S2352340915001626?via%3Dihub#bib1)** about a data set and algorithm for counting red blood corpuscles.

The images we are starting with are stored in BMP format.



We have already taken care of the first steps, so you will be working with a text file that has already been segmented. We have taken the above image, and produced a black and white images where all pixels have the values of 0 (black) or 255 (white).



We then converted the file to ASCII (plain text) format so that you can read it using `fscanf().` (Read the man page for `fscanf()` by typing man fscanf at the command line) You can download **A2.txt** and look at

it in a text editor.

The file contains only integers separated by a space, and you can assume that the file format is correct. The first two integers (first line) are the width and height of the image in pixels. There is a newline character after each row of pixels to make it slightly easier to visually inspect the file. *(Note: The very first version of A2.txt that I uploaded had the first two numbers reversed. The corrected version stores height and then width.)*

Step 0: Set up

Your program will consist of two files: `count_cells.c` and `image.c`.

These files have already been added to you repo. You should be able to compile your program using the following gcc line:

```
gcc -Wall -g -std=gnu99 -o count_cells count_cells.c image.c
```

Step 1: Reading and writing an image file

The main function in `count_cells.c` takes one or two command-line arguments: - The first argument the name of blood cell image in the text file format described above. The second argument, "`-p`" is optional. If present, this argument indicates that the pixel matrix should be printed after processing but before other program output. The program should print the following message and call exit with a value of 1 if there are two many or too few arguments, or if the second argument is not "-p".

```
"Usage: count_cells <imagefile.txt> [-p]"
```

In image.c you will implement three functions. The first two function prototypes are given below (and in the starter code):

```
void read_image(int num_rows, int num_cols, int arr[num_rows][num_cols], FILE *fp);

void print_image(int num_rows, int num_cols, int arr[num_rows][num_cols]);
```

Because we are passing in a variable-length, multi-dimensional array as an argument to a function, the first two arguments must be the dimensions of the array, and the third the array itself. (See King pages 197-199) We could call `read_image` as

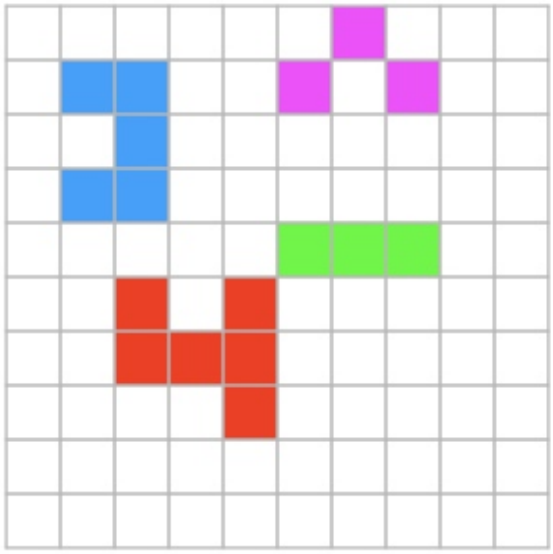
```
int p, q; // assume p and q have valid values
int a[p][q];
FILE *fp; // assume fp holds a valid open file pointer
read_image(p, q, a, fp);
```

This means that we need to read the dimensions of the array before calling read image. Write these two functions and call them from the main function in `count_cells.c` code and test it before moving on. Remember to add the prototypes to `count_cells.c` This would be a good time to learn the basics of gdb.

Step 2: Count cells

Write a function `count_cells` to count the number of blood cells in the image. One cell is a region of adjacent white pixels. The function will be implemented in `image.c`

A pixel is adjacent to another pixel if there is pixel of the same colour above, below, left or right of the pixel In the example image below there are 6 cells. The "blue cell" is composed of 5 pixels, the red cell is composed of 6 pixels, the green cell is composed of 3 pixels (and has no pixels adjacent to the red cell), and there are three pink cells of one pixel each.



The main function will call `count_cells` and will print a message to stdout using the following format string: `"Number of Cells is %d\n"`

Hint: You can modify the 2d array of pixels.

Input

You probably want to create a few smaller image files so that you can verify that your algorithms are working correctly, and so that you can test some edge cases. For example, here is one that I wrote: (I will leave it to you to figure out how many cells are in this image)

```
4 7
0 0 255 255 0 0 0
0 255 0 0 255 255 0
0 255 0 0 0 0 0
255 255 0 0 0 0 0
```

There are a number of `.bmp` files and the resulting .txt files in `/u/csc209h/fall/pub/a1` on teach.cs that you can also use as input. Note that these aren't that useful for testing, unless you want to count by hand the number of cells in them.

What to submit

Add, commit, and push your `image.c` and `count_cells.c` file to your repository for a1.