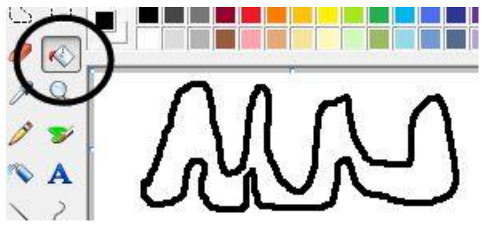
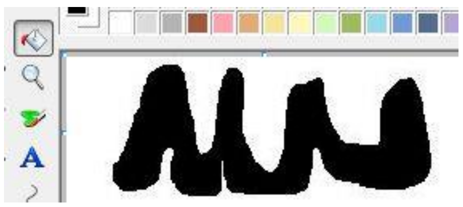
Application of a Stack and Recursion

Many painting programs allow you to 'Fill Color' with a paint tool so that a figure with a white background can be filled with a solid black color. In order to do that, often you can select the "paint can tool", the colour, and then a point in the canvas. And then the silhouetted figure is flooded with colour, staying within the black boundaries.

For instance, the figure



is filled into



For this question, you are to write two versions of the fillBlack function that takes three arguments: a 2D array of type PixelValues and a position (i, j).

```
enum PixelValues {BLACK, WHITE};
const int ROWS = 10;
const int COLS = 20;
```

Use the function prototype for Question 2 and for Question 3

```
void fillBlack(PixelValues canvas[][COLS], int i, int j);
```

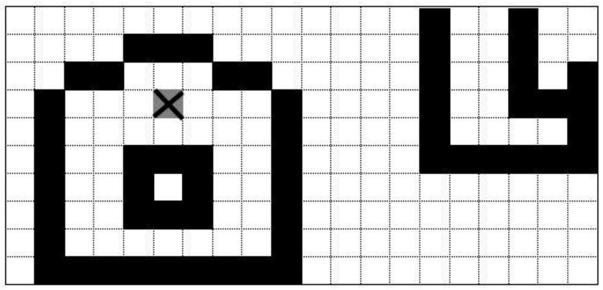
The purpose of fillBack can be described as follows.

If the cell at (i, j) is WHITE, then fillBlack sets the cell (i,j) to BLACK and it sets to BLACK as well all the vertical and horizontal WHITE neighbours. It continues setting WHITE cells to BLACK as long as the WHITE cells are connected vertically or horizontally and are within a BLACK boundary and within the canvas' boundary.

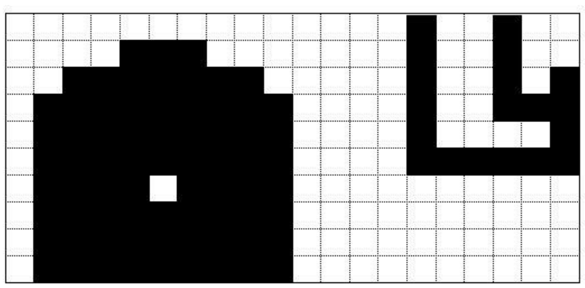
If the cell (i, j) is BLACK already, then fillBlack does not do anything.

The upper left hand corner cell of the canvas is (0,0) and the dimensions of the canvas are ROWS by COLS which are globally accessible named constants.

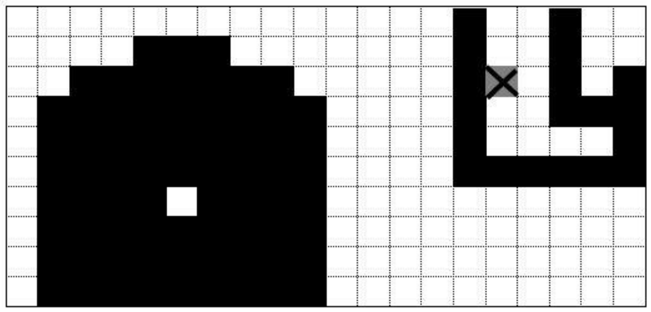
In the canvas below, the starting cell is (3, 5) and it is marked in grey with an x on it:



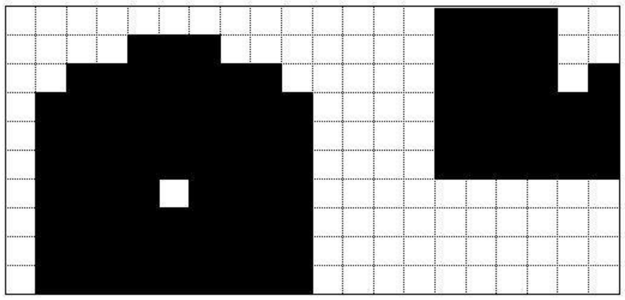
and the result of fillBlack at (3, 5) is



If we call fillBlack again, and this time we give the starting cell (2, 15) marked with an x



then the resulting canvas is



In the canvas array in general, if canvas[x][y] is BLACK it means that the pixel is black and that it is part of the foreground whereas when canvas[x][y] is WHITE, it means that the pixel is part of the background. Modify the canvas passed as an argument to fillBlack by simply assigning new values to the elements canvas[x][y]

Question 1

Implement fillBlack recursively.

Question 2

Implement fillBlack iteratively using an STL stack explicitly.

You can represent the position (i,j) with a single integer encoding as i * COLS + j

Question 3

Implement fillBlack iteratively using an STL queue explicitly.

Question 4

Comment on the order in which the pixels are visited in Question 1, Question 2, Question 3