Test #2 Topics

- Parameter passing
  - value, reference, const reference
- Classes
  - Documentation/Class design
  - Interface vs. implementation
  - Free vs. member functions
- Structs
- Enumerated Types
- Vectors
  - Basic methods
  - Size vs. capacity
  - Searching
  - Sorted insertion & deletion
- Streams
  - Reading by char
  - Reading by line
  - Reading by "word"
  - Reading strings
- Recursion
  - Tracing function calls
  - Base and recursive cases
  - Iteration and tail-recursion
- Algorithmic design
  - Data & loop invariants
  - Patterns
- Matrices
  - Nested loops
- Built-in classes, C++...

Matrices

- Matrix class tmatrix is a 2-dimensional version of a vector
  - Useful for tables, bitmaps, and lots of mathematical operators
  - First index is the row, second index is the column
- Common idiom for iterating through matrix

```cpp
tmatrix<int> mat(rows, cols);
for (int i=0; i < rows; i++)
  for (int j=0; j < cols; j++)
    { // do something with mat[i][j] }
```

Blob Counting: Recursion at Work

- Blob counting is similar to what’s called Flood Fill, the method used to fill in an outline with a color (use the paintcan in many drawing programs to fill in)
  - Possible to do this iteratively, but hard to get right
  - Simple recursive solution
- Suppose a slide is viewed under a microscope
  - Count images on the slide, or blobs in a gel, or …
  - Erase noise and make the blobs more visible
- To write the program we’ll use a class CharBitMap which represents images using characters
  - The program blobs.cpp and class Blobs are essential too

Counting blobs, the first slide

```prompt
prompt> blobs
enter row col size 10 50
# pixels on: between 1 and 500: 200
```

- How many blobs are there? Blobs are connected horizontally and vertically, suppose a minimum of 10 cells in a blob
  - What if blob size changes?
Identifying Larger Blobs

blob size (0 to exit) between 0 and 50: 10

.1111111111
.1111111111
.1111111111
.1111111111
.1111111111
.1111111111
.1111111111
.1111111111
.1111111111
.1111111111

# blobs = 3

- The class Blobs makes a copy of the CharBitMap and then counts blobs in the copy, by erasing noisy data (essentially)
  - In identifying blobs, too-small blobs are counted, then uncounted by erasing them

Identifying smaller blobs

blob size (0 to exit) between 0 and 50: 5

.1111111111
.1111111111
.1111111111
.1111111111
.1111111111
.1111111111
.1111111111
.1111111111
.1111111111
.1111111111

# blobs = 8

- What might be a problem if there are more than nine blobs?
  - Issues in looking at code: how do language features get in the way of understanding the code?
  - How can we track blobs, e.g., find the largest blob?

Recursive helper functions

- Client programs use Blobs: :FindBlobs to find blobs of a given size in a CharBitMap object
  - This is a recursive function, private data is often needed/used in recursive member function parameters
  - Use a helper function, not accessible to client code, use recursion to implement member function

- To find a blob, look at every pixel, if a pixel is part of a blob, identify the entire blob by sending out recursive clones/scouts
  - Each clone reports back the number of pixels it counts
  - Each clone “colors” the blob with an identifying mark
  - The mark is used to avoid duplicate (unending) work

Conceptual Details of BlobFill

- Once a blob pixel is found, four recursive clones are “sent out” looking horizontally and vertically, reporting pixel count
  - How are pixel counts processed by clone-sender?
  - What if all the clones ultimately report a blob that’s small?

- In checking horizontal/vertical neighbors what happens if there aren’t four neighbors? Is this a potential problem?
  - Who checks for valid pixel coordinates, or pixel color?
  - Two options: don’t make the call, don’t process the call

- Non-recursive member function takes care of looking for blobsize, then filling/counting/unfilling blobs
  - How is unfill/uncount managed?
What's a pixmap, a bit, a byte, a char?

- The pbm ASCII format we're using in Pixmap assignment is not for conserving space, it's for ease-of-reading
  - Here's a 3x6 (rows,columns) image, what is it?
    0 0 0 0 1 1 1 1 1 0 0 0 0 0 0
  - How much disk space does this use?
    • (18 + 18) chars/spaces * 8 bits/char = 288 bits = 36 chars
    • We have 18 bits, should be able to store in 4 chars
- The xv toolkit can save an image in raw pbm too
  - Uses one bit per 0/1 rather than 2 bytes (space)
  - To convert back, read a byte/char at a time, and peel off the bits, see convertpbm.cpp

00101010 = 0x27 + 0x26 + 1x2^5 + 0x2^4 + 1x2^3 + 0x2^2 + 1x2^1 + 0x2^0