PA9 – ExtraCredit: Boustrophedonic Lists

Due Date

- See Piazza for any changes to due date and time
  - Friday by midnight
  - Grading the next day Saturday Morning
- Submit program to perforce in your student directory
  - Sub directory called:
    - /PA9 /...
  - Fill out your ***PA9_ Submission Report.pdf***
    - Place it in the /PA9 directory
    - Enter the final Changelist number of your submission
    - Write up a quick discussion in the report
      - What you learned from this assignment

Goals

- Programming Assessment
  - 5% Extra credit for PAs - More practice with linked lists... (kind of)

Assignments

- Write a single function: remove()
  - Signature and structure need exactly the same
  - Place function in file named (see supplied files):
    - Boustrophedonic.cpp:
      - Function
    - Boustrophedonic.h:
      - Declaration and structure

```
struct Node
{
    Node *pNorth;
    Node *pSouth;
    Node *pEast;
    Node *pWest;
}

void Remove( Node *&head, int row, int col );
```

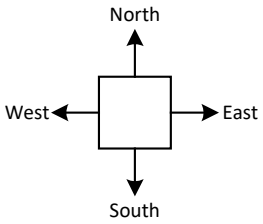
- Assumptions
  - Boustrophedonic list is complete and ***PRISTINE*** before you remove one node
  - Only ***ONE*** node will be removed in the function
  - The dimensions of the list are ***NOT*** given
  - Boustrophedonic list has an even number of columns
  - Boustrophedonic list has no restrictions on number of rows
  - On deletion, horizontal and vertical connections are preserved across the deleted node

Creating a Boustrophedonic List

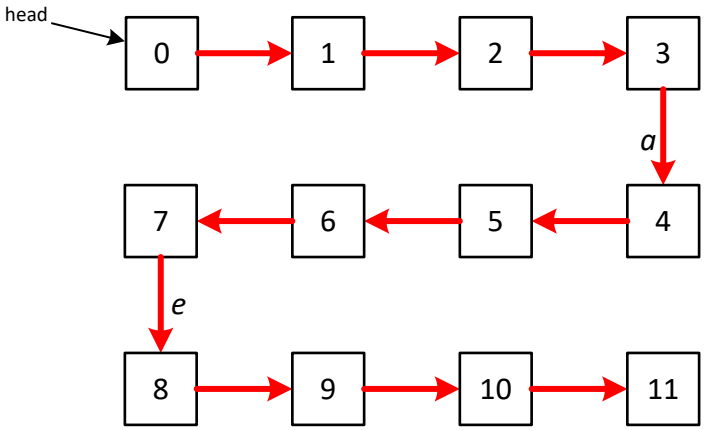
Boustrophedonic - Greek word:

- A method of writing shown in early Greek inscriptions, in which the lines run alternately from right to left and from left to right, as the furrows made in plowing a field, the plow passing alternately backward and forward.

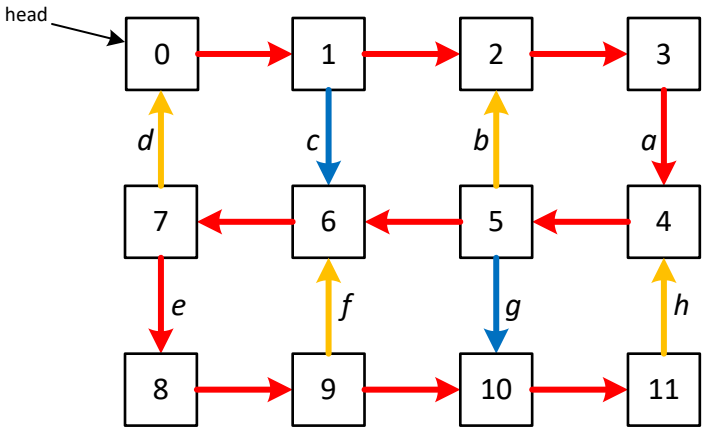
```
struct Node
{
    Node *pNorth;
    Node *pSouth;
    Node *pEast;
    Node *pWest;
}
```



- Start with the first node and continue in the Boustrophedonic path way, follow the RED LINE.
    - Start with the head, and sequentially go from 0 to 11 in the Boustrophedonic way

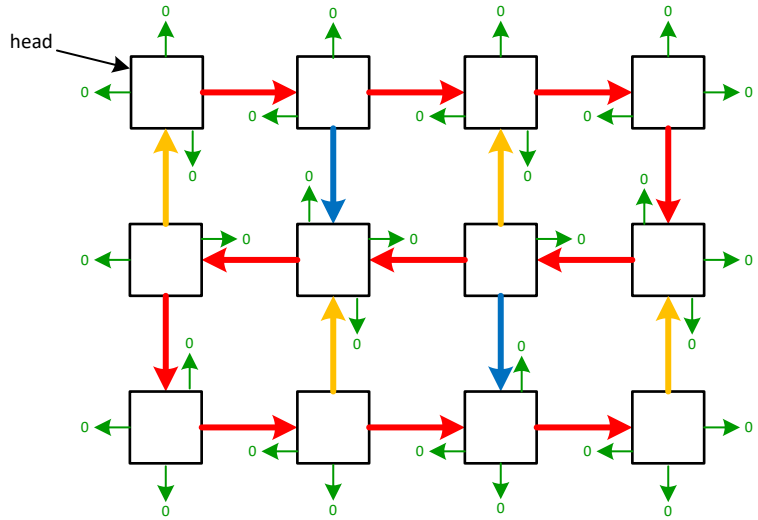


- Next we create the vertical connections.

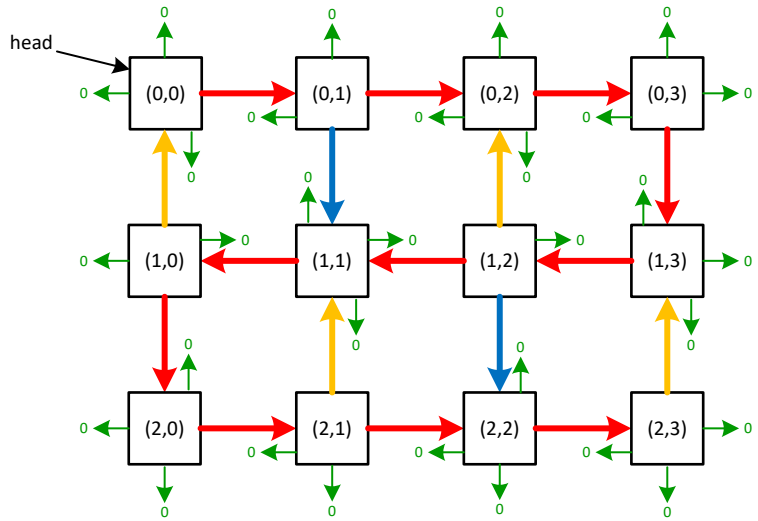


- Create the vertical connections between row 0 {0,1,2,4} and row 1 {7,6,5,4}.
  - Find arrow *a* between 3 and 4.
    - That arrow goes down, from 3 to 4.
  - Alternate direction of *a*, (*a* goes down in RED)
    - *b* goes up YELLOW,
    - *c* goes down BLUE,
    - *d* goes up YELLOW.
- Similar for the connections between row 1 {7,6,5,4} and row 2 {8,9,10,11}
  - Alternate direction of *e*, (*e* goes down in RED)
    - *f* goes up YELLOW,
    - *g* goes down BLUE,
    - *h* goes up YELLOW.

- Drawing the list with all the links displayed.
  - Each node actually has 4 pointers, most are nulls.
    - North, South, East, West
  - Here the list is drawn with GREEN links showing the explicit null pointers.

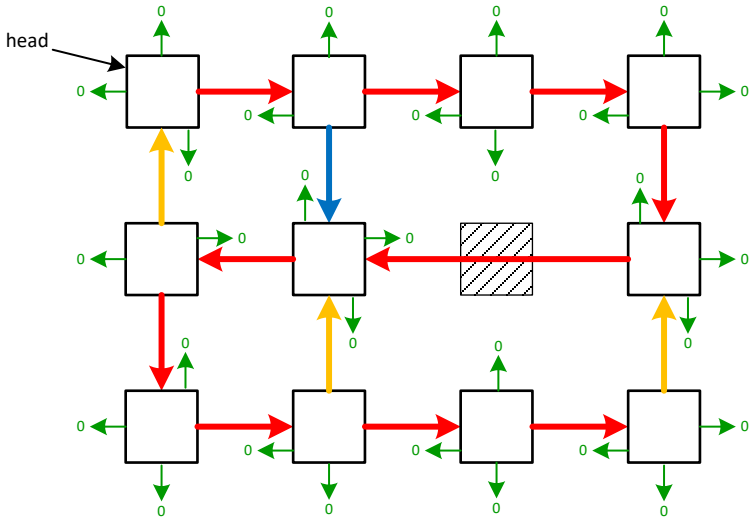


- Rows and Columns
  - Rows and columns are not provided, nor are they added in the Node structure.
  - You need to calculate these as you walk the list.
  - The lists will vary in dimensions, Row x Column (row,col)
  - There will be an even number of columns.

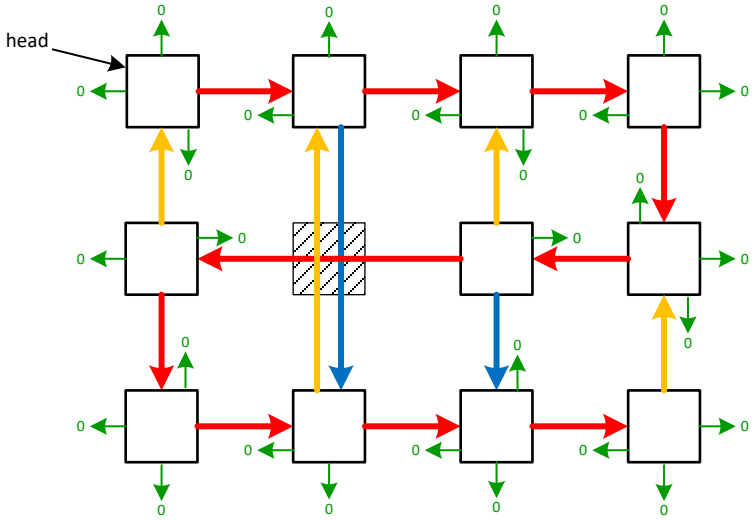


- Deletion
  - Remove only one node at a time
  - Keep horizontal connections, through the deleted node
    - Fix adjacent nodes connections
  - Keep vertical connections, through the delete node
    - Fix adjacent nodes connections
  - Many examples to follow

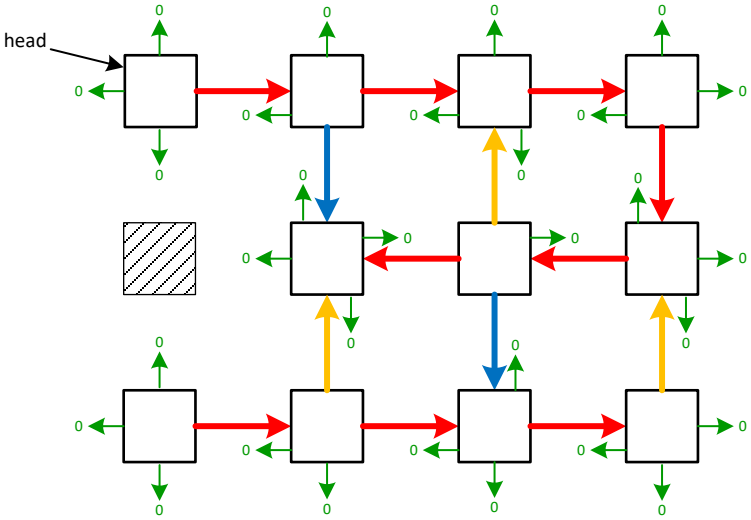
Delete Node (1,2)



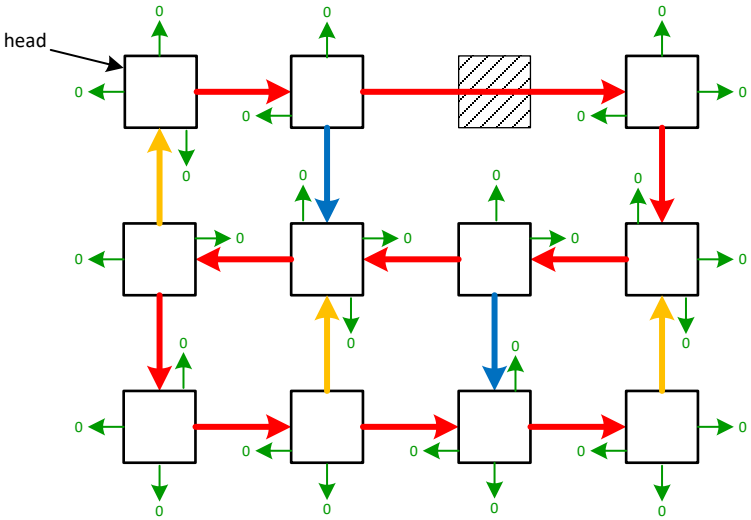
Delete Node (1,1)



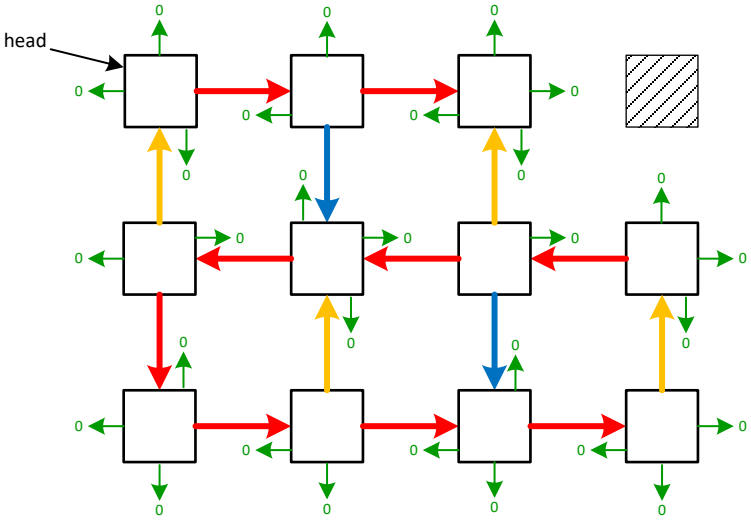
Delete Node (1,0)



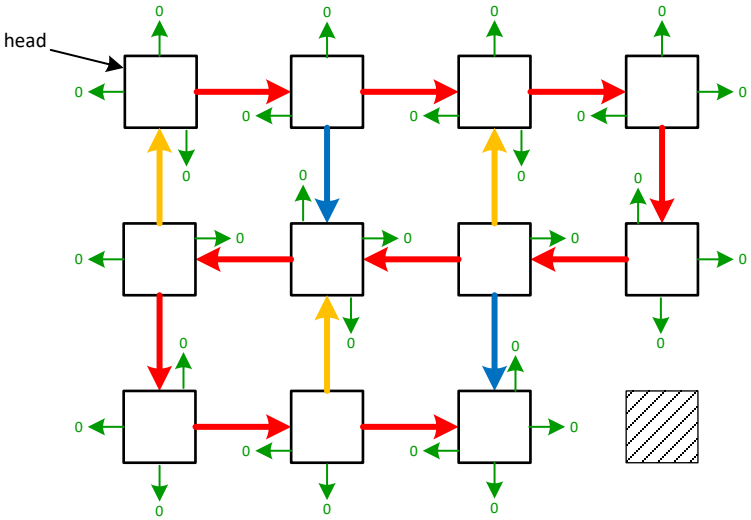
Delete Node (0,2)



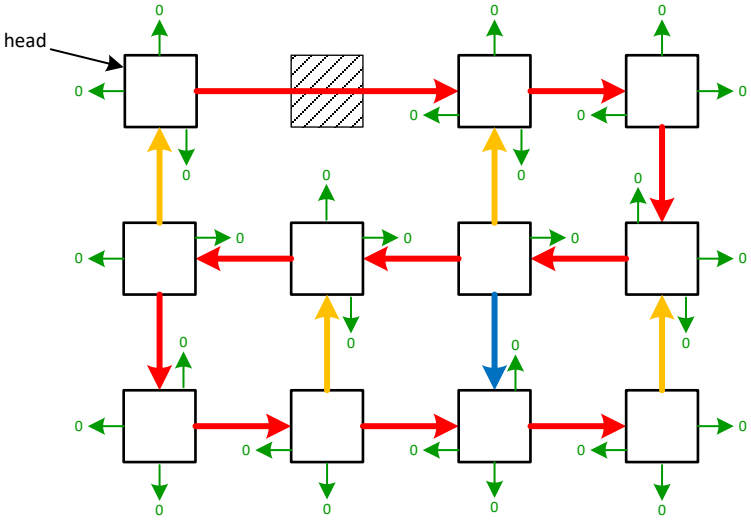
Delete Node (0,3)



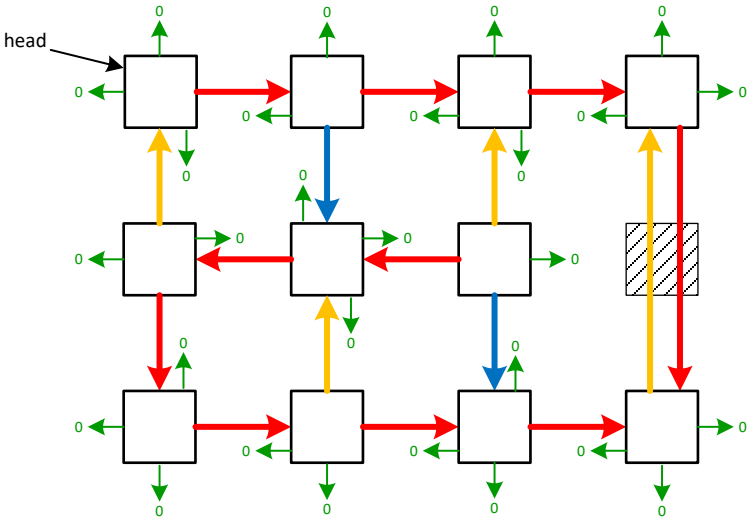
Delete Node (2,3)



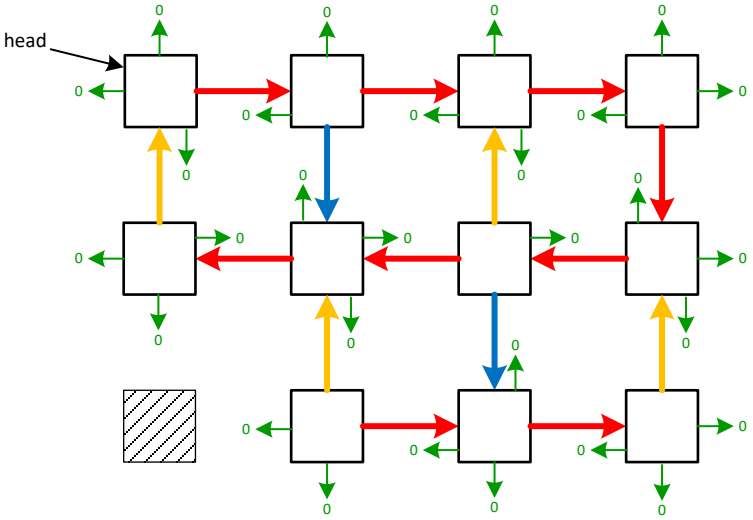
Delete Node (0,1)



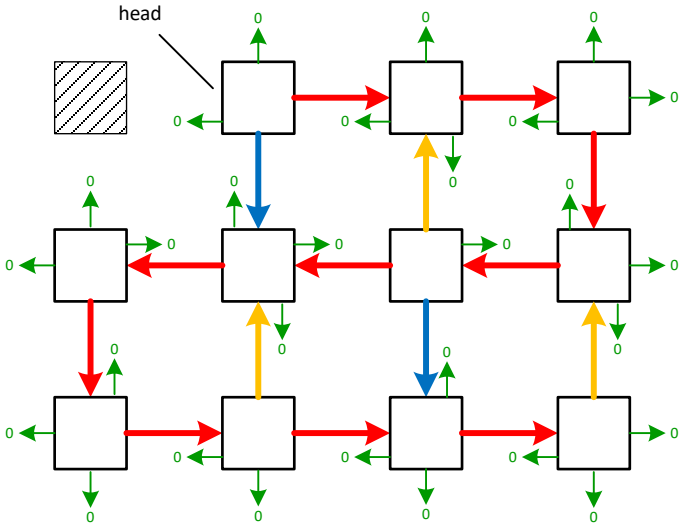
Delete Node (1,3)



Delete Node (2,0)



Delete Node (0,0)



General:

- Write all programs in cross-platform C or C++.
  - Optimize for execution speed and robustness.
- Create a programming file for each problem, for example
  - Student directory
    - /PA9/…
  - Make sure that each problem can be compiled and run through the checked in solution
- Write all programs in cross-platform C or C++.
  - Optimize for execution speed and robustness.

- Do all your work by yourself
  - Feel free to talk with others about ideas on Piazza
  - You are 100% responsible for all code
  - See syllabus about collaboration rules
- Fill out the submission Report
  - No report, no grade
- We are using Perforce
  - You should have received the document describing how to login.
  - Please look at the documentation and videos under the reference directory
  - All your code must compile from perforce with no modifications.
    - Otherwise it's a 0, no exceptions
  - Only Visual Studio 2015 allowed
- Submit your work as you go to perforce several times (at least 3)
  - As soon as you get something working, submit to perforce
  - Have reasonable check-in comments
    - Seriously, I'm checking
- Make sure that your program compiles and runs
  - Warning level ALL sometimes that is not possible due to MS headers…
    - There are corrections around windows headers
  - NO Warnings or ERRORS
    - Your code should be squeaky clean.
- NO Boost
- NO STL allowed {Vector, Lists, Sets, etc...}
  - No automatic containers or arrays
    - You need to do this the old fashion way - ***YOU EARNED IT***
- No modern C++
  - Lambdas, Autos, templates, etc…

Validation

Simple check list to make sure that everything is checked in correctly

- Do they compile and run without any errors?
- Warning level All free?
- Submitted it into /PA9 directory?
- Did you verify the submission?

Hints

Most assignments will have hints in a section like this.

- Do many little check-ins
  - Iteration is easy and it helps.
  - Perforce is good at it.