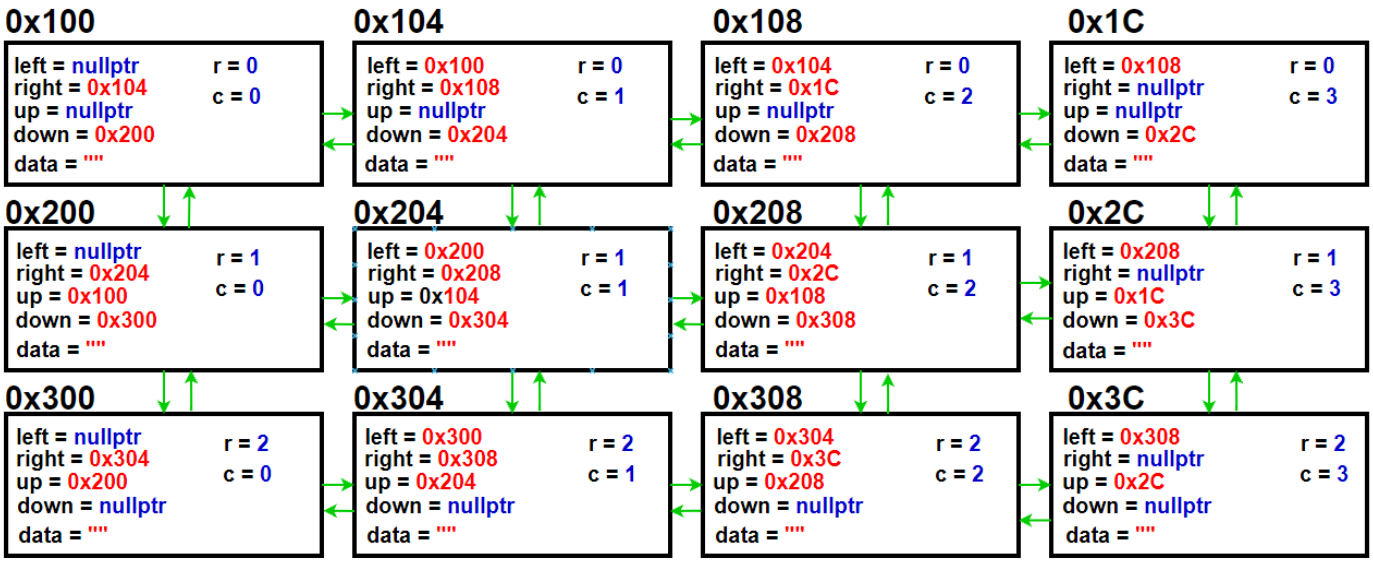
DSA - PROJECT
Creating a Mini-Ultimate Excel Sheet Manager

Your goal is to create a class called mini excel. This class will have a subclass of *cell* . The major concept to be used is the same as a doubly linked list as we have already implemented. In the doubly linked list, each node had two links (next and previous). Think of each cell in excel sheet as a node. Each node will now have four links (up, down, left and right). The top left cell of the sheet will be treated as the root node.

All cells in the top most row will have *nullptr* in their *up* link.
All cells in the last row will have *nullptr* in their *down* link.
All cells in the right most column will have *nullptr* in their *right* link.
All cells in the left most column will have *nullptr* in their *left* link.
You can get a better understanding of the links from the figure below:



Functions to be included in Mini Excel Class:

1. **Excel()**
   This is the constructor of your Excel class. It initially makes a 5 by 5 grid of cells having space character as data. We keep a cell* (call it current) as an attribute of Excel class. This is actually the active cell of your excel sheet..
2. **InsertRowAbove()**
   Inserts a row above the current cell. You will have to play with *up* links of all cells in the given row index. Set the four links of newly inserted cells accordingly.
3. **InsertRowBelow()**
   Inserts a row below the current cell. You will have to play with the downlink of all cells in the given row index. Set the four links of newly inserted cells accordingly.
4. **InsertColumnToRight()**
   Inserts a column to the right of the current cell. You will have to play with the right link of all cells in the given column index. Set the four links of newly inserted cells accordingly.

5. **InsertColumnToLeft()**
   Inserts a column to the left of the current cell. You will have to play with the left link of all cells in the given column index. Set the four links of newly inserted cells accordingly.
6. **InsertCellByRightShift()**
   It shifts the current cell to the right and adds a new cell in its place. Note that this function will result in insertion of a new column at the rightmost end of the sheet.
7. **InsertCellByDownShift()**
   It shifts the current cell one row down and adds a new cell in its place. Note that this function will result in insertion of a new row at the bottom end of the sheet.
8. **DeleteCellByLeftShift()**
   It deletes the current cell and left shifts all cells to its right (only in the deleted cell's row) by one step. Update current cell to be the cell which was to the left of deleted cell.
9. **DeleteCellByUpShift(l)**
   It deletes the current cell and shifts all cells below it (only in the deleted cell's column) one step up. Update current cell to be the cell which was above the deleted cell.

10. **DeleteColumn()**
    It deleted the column of the current cell. It then updates the links of involved cells accordingly.
11. **DeleteRow()**
    It deletes row of the current cell.It then updates the links of involved cells accordingly.
12. **ClearColumn()**
    It clears data of the entire column of current cell. When we display the sheet, the cleared cells should be shown empty.
13. **ClearRow()**
    It clears data of the entire row of the current cell. When we display the sheet, the cleared cells should be shown empty.
14. **GetRangeSum(RangeStart, RangeEnd)**
    It receives indices of starting and ending cells in the range and computes the range sum. Refer to MS Excel Spreadsheet to learn how range sum works.
15. **GetRangeAverage(RangeStart, RangeEnd)**
    It receives indices of starting and ending cells in the range and computes the range average. Refer to MS Excel Spreadsheet to learn how range average works.
16. **Iterator class**
    The iterator should be able to move up, down, left and right. You can use pre-increment and pre-decrement operators for row increment and row decrement. Similarly, post increment and post decrement operators can be used for column increment and column decrement.
17. **PrintSheet()**
    It should print all the contents of the sheet in tabular form. You will have to make PrintGrid(), PrintCellBorder(), PrintDataInCell() functions as well.

18. **Implement the following functions. All of these take a range starting and range ending cell.**

    SUM
    AVG
    COUNT
    MIN
    MAX

    The result should be printed in the cell selected by the user.

19. **Copy()**

    It selects a cell range and stores the data in cells in some array/ vector.

20. **Cut()**

    It performs the function of Copy() with an additional task of clearing the data in selected cells.

21. **Paste()**

    It pastes the data of cut/ copied cells in the cells starting from the current cell. If there are not enough cells after the current cell, it adds more rows/ columns accordingly.

22. **Cell Functions**

    Your cell class should have the following attributes:
    Color
    Cell alignment
    Data type(char,int or float)
    Data stored in cell should be string of length 4

23. **File Handling**

    Create sheet save and sheet load functions.

24. **Bonus**

    Resetting of cell width and height on inserting characters more than cell width as explained in class.

Good luck