INFORMATION TECHNOLOGY UNIVERSITY

Data Structures and Algorithm

Lab 08

Instructor	Shoaib Majeed
Teaching Assistants	Amna Amir bscs22059@itu.edu.pk Abdul Moqeet bscs22147@itu.edu.pk
Semester	Fall 2024

Department of Computer Science ITU, Lahore
Pakistan

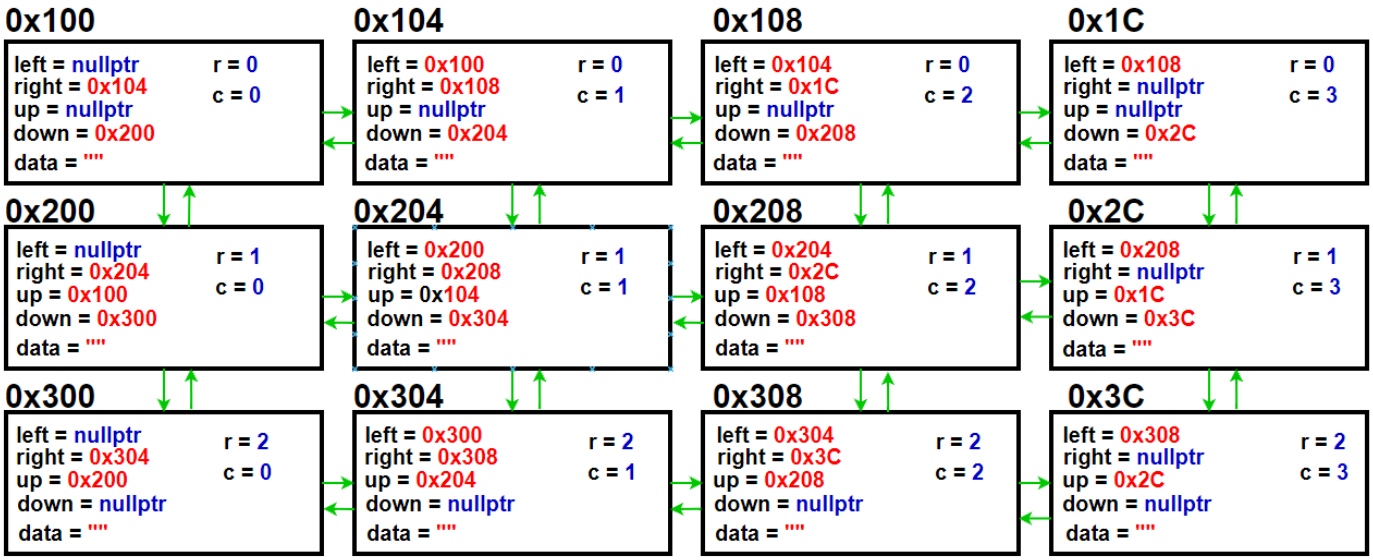
Note: Submit a zip file with naming convention bscs23xxx_MiniExcel.zip

Mini Excel

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. 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 the 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. **DeleteColumn()**
   It deleted the column of the current cell. It then updates the links of involved cells accordingly.
7. **DeleteRow()**
   It deletes the row of the current cell.It then updates the links of involved cells accordingly.
8. **ClearColumn()**
   It clears data of the entire column of the current cell. When we display the sheet, the cleared cells should be shown empty.
9. **ClearRow()**
   It clears data of the entire row of the current cell. When we display the sheet, the cleared cells should be shown empty.
10. **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.
11. **PrintSheet()**
    It should print all the contents of the sheet in tabular form. You will have to make PrintGrid(), PrintCellBorder(), PrintDataInCell() functions as well.

Bonus:

12. **Copy()**
    It selects a cell range and stores the data in cells in some array/ vector.
13. **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.

Prototype:

```
#include<iostream>
#include<vector>
#include<iomanip>
using namespace std;

class MiniExcel {
private:

    class Cell {
    public:
        char data;   // Content of the cell
        Cell* up;    // Link to the cell above
        Cell* down;  // Link to the cell below
        Cell* left;  // Link to the cell to the left
        Cell* right; // Link to the cell to the right

        // Write the Constructor to initialize a cell with given data (default space '
')
        Cell(_____) {}
    };

    Cell* root;      // Root cell (top-left cell)
    Cell* current;   // Current active cell
    int rows;        // Number of rows in the grid
    int cols;        // Number of columns in the grid

public:
    // Constructor to initialize a 5x5 grid with space characters
```

```
MiniExcel();

// Insert a row above the current cell
void insertRowAbove();

// Insert a row below the current cell
void insertRowBelow();

// Insert a column to the right of the current cell
void insertColumnToRight();

// Insert a column to the left of the current cell
void insertColumnToLeft();

// Delete the current column
void deleteColumn();

// Delete the current row
void deleteRow();

// Clear data of the entire column of the current cell
void clearColumn();

// Clear data of the entire row of the current cell
void clearRow();

// Iterator class for moving up, down, left, right
class Iterator {
private:
    Cell* currentCell;

public:
    // Constructor initializing the iterator to a given cell
    Iterator(_____) {}

    // Move the iterator up
    Iterator& operator--();  // Pre-decrement (move up)
```

```
    // Move the iterator down
    Iterator& operator++();  // Pre-increment (move down)

    // Move the iterator left
    Iterator operator--(int);  // Post-decrement (move left)

    // Move the iterator right
    Iterator operator++(int);  // Post-increment (move right)
};

// Get an iterator to the current cell
Iterator getIterator();

// Print the entire sheet
void printSheet()
{
	Cell* rowPtr = root;
	while (rowPtr != nullptr)
	 {
		Cell* colPtr = rowPtr;
		while (colPtr != nullptr)
		{
			cout << setw(3) << colPtr->data << " | ";
			colPtr = colPtr->right;
		}
		cout << endl;
		Cell* temp = rowPtr;
		while (temp != nullptr)
		{
			cout << "-----";
			temp = temp->right;
		}
		cout << endl;
		rowPtr = rowPtr->down;
	}
}
```

```
    // Bonus:
    // Copy a selected cell range to a vector
    vector<char> copy(int startRow, int startCol, int endRow, int endCol);

    // Paste data starting from the current cell
    void paste(const vector<char>& data,int startRow, int startCol);
};
```