Programming Assignment #2

資料結構：

這次的程式設計中使用了以下的結構

1. struct Row：

   儲存 row 的基本資訊，如 row 名稱、位置、方向、site 數量與間距。

```
struct Row {
    string rowName;
    string siteName;
    int x, y;
    string orientation;
    int numX, numY;
    int stepX, stepY;

    Row(string rowName, string siteName,int x,int y, string orientation, int numX, int numY, int stepX, int stepY);
};
```

2. struct Component：

   每個元件包含原始位置與合法化後的位置（laterX, laterY）、orient 方向與 displacement。

```
struct Component {
    string instName;
    string macroName;
    int originalX, originalY;
    int laterX, laterY;
    string orientation;
    int distance;

    Component(string instName, string macroName, int originalX, int originalY, string orientation);
};
```

3. struct Cluster：

   在後處理階段進行局部最佳化時，將 displacement 較大的 cell 以一個 cluster 的形式收集，並記錄所有對應 site。

```
struct Cluster {
    vector<Component*> cells;
    vector<int> sites;
    int originalMaxDisp;
};
```

4. struct PermResult：

   記錄 cluster 的排列結果與對應的最大 displacement，用來判斷 permutation 效果。

```
struct PermResult {
    vector<Component*> order;
    int maxDisp;
};
```

演算法：

整體會分為三個流程

1. 預估每個 cell 的合法位置 (FindCellBestPlace)
   根據 cell 原始位置粗略對應到最接近的 site，計算初步的合法化位置與 displacement。
2. Greedy Placement (CellPlace)
   因為我前面已經有做距離的排序，所以這個步驟會先去擺放距離最遠的那個 cell 以減少 max distance 的影響，會把每個 cell 去跟每個 site 去判斷是否可以放置，會找到沒有被放置且 distance 最小的位置。
3. Top-K Cluster 優化
   會有這個演算法的出現是因為我有同學做 abacus，我們的 average distance 是差不多的，不過他的 max distance 比我小很多，所以我才會想要去優化我的 max distance，這個步驟核心是會找出距離最大的前 6 個 cell，然後去找出同一個 row 中的左右各 5 個 cell 去做所有可能的排序，看這個群組甚麼排法的 max distance 最小，不斷重複 50 次。
   就結果來說在 super1 中的測試讓我的 max distance 下降了 40%