資料結構

Data Structure

Lab 09

姓名：曾致嘉

學號：113AB0014

Lab09-Q1

Please implement the missing heapify function for the Max Heap in the sample code, ensuring that both Max Heap and Min Heap work correctly. Then, read the three input files (input1.txt, input2.txt, and input3.txt) to test your output results. An executable file is provided so you can verify your results against the correct outputs.

Code

```
#include <iostream>
#include <vector>
#include <fstream>
#include <sstream>
using namespace std;

// 從文件中讀取數據並存入向量
vector<int> readFromFile(const string& filename) {
    vector<int> arr;
    ifstream file(filename);

    if (!file) {
        cerr << "Error opening file: " << filename << endl;
        return arr;
    }

    string line;
    while (getline(file, line)) {// 持續讀取整行內容
        stringstream ss(line);      // 創建字符串流
        string value;
        while (getline(ss, value, ',')) {// 用逗號分隔值
            try {
                arr.push_back(stoi(value)); // 將字符串轉換為整數並存入向量
            }
            catch (exception& e) {
                cerr << "Invalid number format in file: " << value << endl;
            }
        }
    }
```

```
    file.close();
    return arr;
}

class MaxHeap {
public:
    vector<int> heap; // 儲存 Max Heap 的元素

    // 建立 Max Heap
    void buildMaxHeap(vector<int>& arr) {   //建立 Max Heap
        heap = arr;

        for (int i = (heap.size() / 2) - 1; i >= 0; i--) { // 從最後一個非葉子節點開始向上執行 Max Heap
            heapify(i);
        }
    }

    void heapify(int i) {// 堆化函式（確保以 i 為根的子樹符合 Max Heap 性質）
        int largest = i;            // 假設當前節點是最大的
        int left = 2 * i + 1;       // 左子節點索引
        int right = 2 * i + 2;      // 右子節點索引

        // if 判斷式，檢查左子節點是否為有效範圍且比當前節點(父節點)大
        //如果判斷式成立，把 largest 設為 left
        if(left<heap.size() && heap[largest]<heap[left])largest = left;

        // if 判斷式，檢查右子節點是否為有效範圍且比當前節點(父節點)大
        //如果判斷式成立，把 largest 設為 right
        if(right<heap.size() && heap[largest]<heap[right])largest = right;

        // if 判斷式，如果最大的不是父節點，交換並繼續堆化
        // 遞迴處理受影響的子樹
        if (largest != i){
            swap(heap[i], heap[largest]);
            heapify(largest);
        }
    }
```

```
    // 顯示 Heap 的內容(使用 BFS)
    void printHeap() {
        for (int val : heap) {// 遍歷 Max Heap 中的每個元素
            cout << val << " ";// 輸出元素
        }
        cout << endl;
    }
};

class MinHeap{
    public:
    vector<int> heap; // 儲存 Mix Heap 的元素

    // 建立 Max Heap
    void buildMinHeap(vector<int>& arr) {   //建立 Mix Heap
        heap = arr;

        for (int i = (heap.size() / 2) - 1; i >= 0; i--) { // 從最後一個非葉子節點開始向上執
行 Mix Heap
            heapify(i);
        }
    }

    void heapify(int i) {// 堆化函式（確保以 i 為根的子樹符合 Mix Heap 性質）
        int smallest = i;               // 假設當前節點是最小的
        int left = 2 * i + 1;       // 左子節點索引
        int right = 2 * i + 2;      // 右子節點索引

        // if 判斷式，檢查左子節點是否為有效範圍且比當前節點(父節點)小
        //如果判斷式成立，把 largest 設為 left
        if (left < heap.size() && heap[left] < heap[smallest])smallest = left;

        // if 判斷式，檢查右子節點是否為有效範圍且比當前節點(父節點)大
        //如果判斷式成立，把 largest 設為 right
        if (right < heap.size() && heap[right] < heap[smallest])smallest = right;

        // if 判斷式，如果最小的不是父節點，交換並繼續堆化
```

```
            // 遞迴處理受影響的子樹
            if (smallest != i){
                swap(heap[i], heap[smallest]);
                heapify(smallest);
            }
        }

        // 顯示 Heap 的內容(使用 BFS)
        void printHeap() {
            for (int val : heap) {// 遍歷 Max Heap 中的每個元素
                cout << val << " ";// 輸出元素
            }
            cout << endl;
        }
};

int main() {
        // 從文件讀取輸入元素
        string filename = "./input1.txt";//請貼上 input 檔案的正確路徑
        vector<int> arr = readFromFile(filename);//讀取數據

        if (arr.empty()) {// 如果數據為空
            cerr << "No valid data found in file." << endl;//輸出錯誤信息
            return -1;
        }
        cout << "Input Array: ";//輸出讀取的數據
        for (int val : arr) {
            cout << val << " ";//輸出每個元素
        }
        cout << endl;

        MaxHeap maxHeap;// 創建 Max Heap 對象
        maxHeap.buildMaxHeap(arr); // 建立 Max Heap

        MinHeap minHeap;
        minHeap.buildMinHeap(arr);

        // 輸出 Max Heap 的內容
```

```
    cout << "Max Heap(By BFS): ";
    maxHeap.printHeap();

    // 輸出 Mix Heap 的內容
    cout << "Min Heap(By BFS): ";
    minHeap.printHeap();
    cout << endl;

    system("pause");
    return 0;
}
```

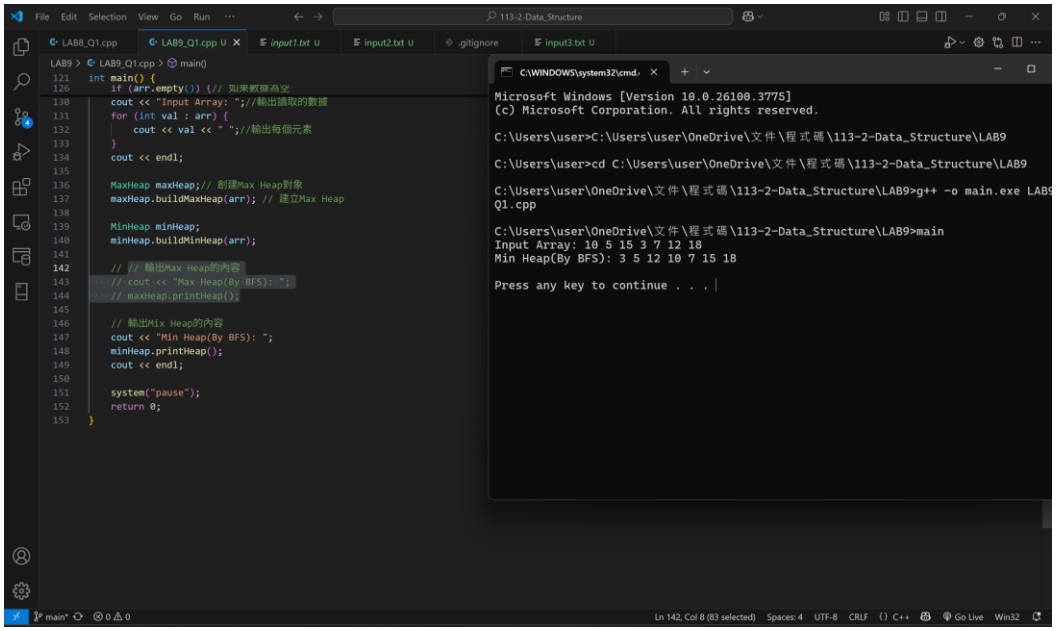
Discussion Section

case 1:

Min：

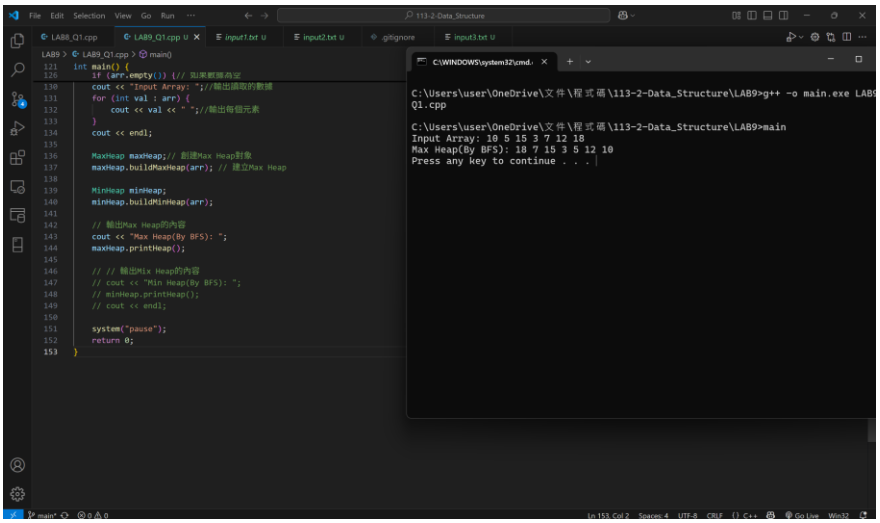
```
C:\Users\user\OneDrive\文件\程式碼\113-2-Data_Structure\LAB9>main
Input Array: 10 5 15 3 7 12 18
Min Heap(By BFS): 3 5 12 10 7 15 18

Press any key to continue . . .
```



Max：

```
C:\Users\user\OneDrive\文件\程式碼\113-2-Data_Structure\LAB9>main
Input Array: 10 5 15 3 7 12 18
Max Heap(By BFS): 18 7 15 3 5 12 10
Press any key to continue . . .
```

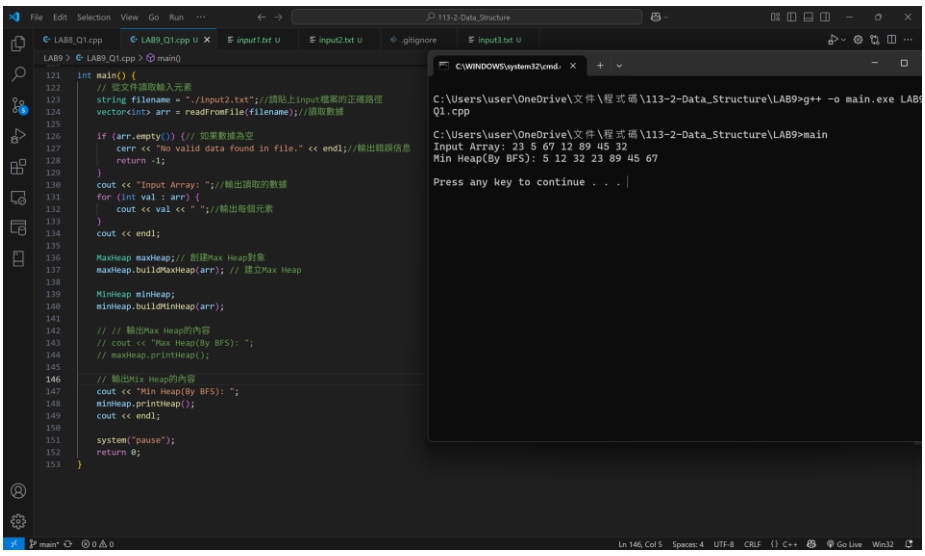


case 2:

Min：

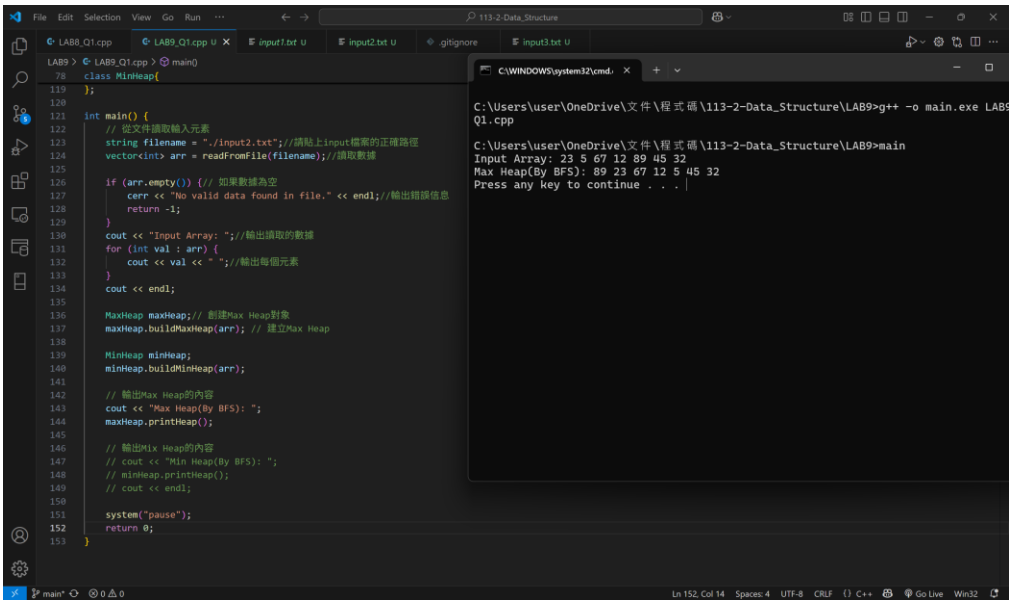
```
C:\Users\user\OneDrive\文件\程式碼\113-2-Data_Structure\LAB9>main
Input Array: 23 5 67 12 89 45 32
Min Heap(By BFS): 5 12 32 23 89 45 67

Press any key to continue . . .
```



Max：

```
C:\Users\user\OneDrive\文件\程式碼\113-2-Data_Structure\LAB9>main
Input Array: 23 5 67 12 89 45 32
Max Heap(By BFS): 89 23 67 12 5 45 32
Press any key to continue . . .
```

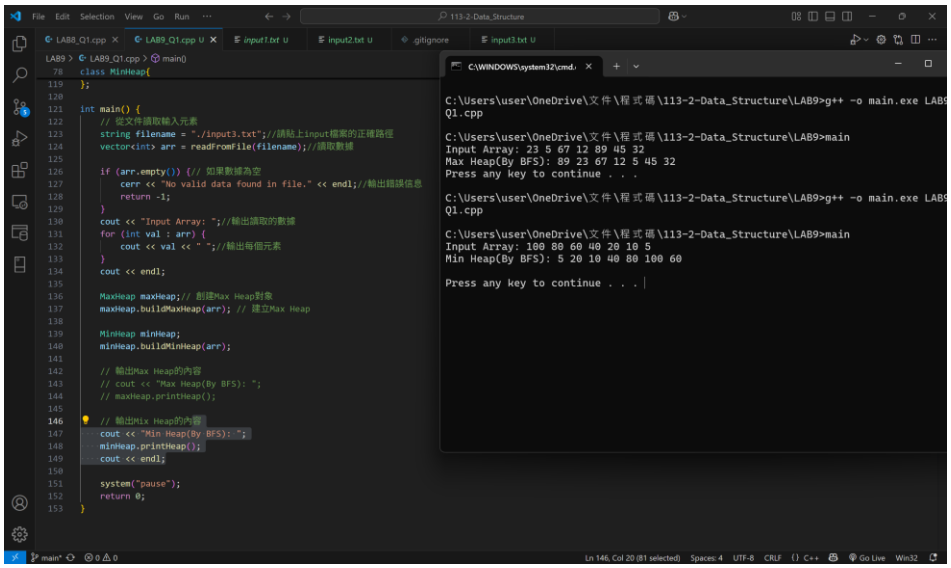


case 3:

Min：

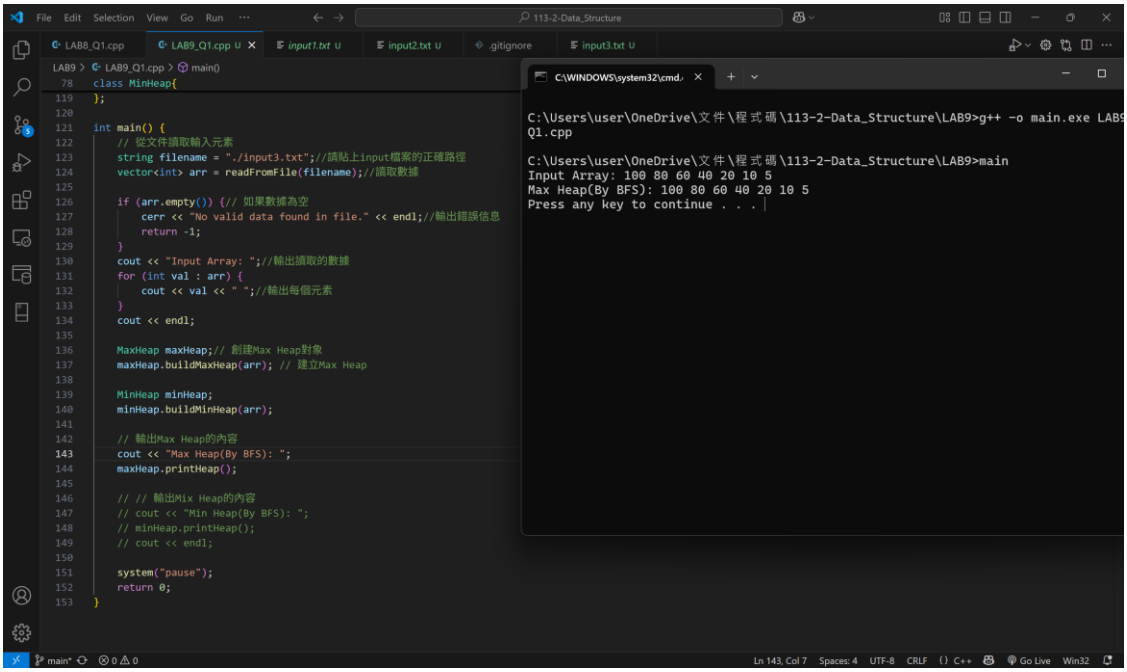
```
C:\Users\user\OneDrive\文件\程式碼\113-2-Data_Structure\LAB9>main
Input Array: 100 80 60 40 20 10 5
Min Heap(By BFS): 5 20 10 40 80 100 60

Press any key to continue . . .
```



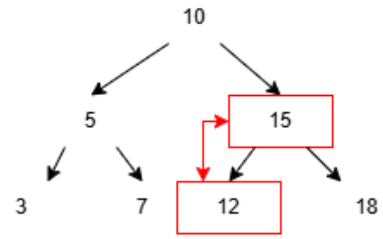
Max：

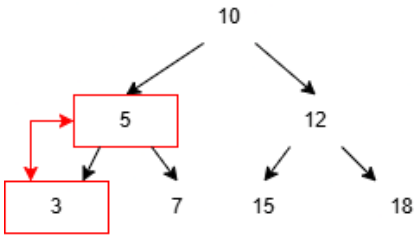
```
C:\Users\user\OneDrive\文件\程式碼\113-2-Data_Structure\LAB9>main
Input Array: 100 80 60 40 20 10 5
Max Heap(By BFS): 100 80 60 40 20 10 5
Press any key to continue . . .
```

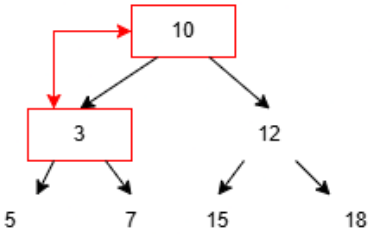


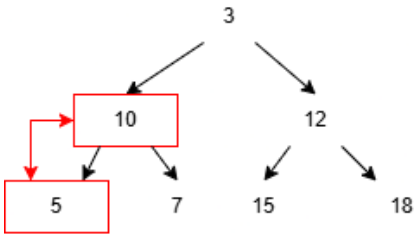
Procedure：

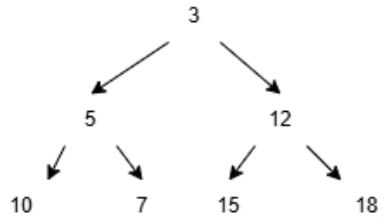
MinHeap











MaxHeap