#include <iostream>

using namespace std;

class employ {

public:

    struct data {

        int id;

        string name;

    };

    data a[100];

    data LeftArray[100], RightArray[100];

    int n;

    void getdata() {

        cout << "Enter the number of employees: ";

        cin >> n;

        for (int i = 0; i < n; i++) {

            cout << "Enter id and name: " << endl;

            cin >> a[i].id;

            cin.ignore();

            getline(cin, a[i].name);

        }

    }

     int partition(int start, int end) {

        int pivot = a[end].id;

        int i = (start - 1);

        for (int j = start; j <= end - 1; j++) {

            if (a[j].id < pivot) {

                i++;

                swap(a[i], a[j]);

            }

        }

        swap(a[i + 1], a[end]);

        return (i + 1);

    }

    void quick(int start, int end) {

        if (start < end) {

            int p = partition(start, end);

            quick(start, p - 1);

            quick(p + 1, end);

        }

    }

    void merge(int beg, int mid, int end) {

        int n1 = mid - beg + 1;

        int n2 = end - mid;

        for (int i = 0; i < n1; i++)

            LeftArray[i] = a[beg + i];

        for (int j = 0; j < n2; j++)

            RightArray[j] = a[mid + 1 + j];

        int i = 0, j = 0, k = beg;

        while (i < n1 && j < n2) {

            if (LeftArray[i].name <= RightArray[j].name) {

                a[k] = LeftArray[i];

                i++;

            } else {

                a[k] = RightArray[j];

                j++;

            }

            k++;

        }

        while (i < n1) {

            a[k] = LeftArray[i];

            i++;

            k++;

        }

        while (j < n2) {

            a[k] = RightArray[j];

            j++;

            k++;

        }

    }

    void mergeSort(int beg, int end) {

        if (beg < end) {

            int mid = beg + (end - beg) / 2;

            mergeSort(beg, mid);

            mergeSort(mid + 1, end);

            merge(beg, mid, end);

        }

    }

    void sortdata() {

        quick(0, n - 1);

        cout << "The Sorted List by ID: " << endl;

        for (int i = 0; i < n; i++) {

            cout << a[i].id << "  " << a[i].name << endl;

        }

    }

    void msort() {

        mergeSort(0, n - 1);

        cout << "The Sorted List By Name: " << endl;

        for (int i = 0; i < n; i++) {

            cout << a[i].id << "  " << a[i].name << endl;

        }

    }

};

int main() {

    char type = 'y';

    int ss;

    employ s;

    s.getdata();

    while (type == 'y') {

        cout << "1) Sort by id" << endl;

        cout << "2) Sort by name" << endl;

        cin >> ss;

        switch (ss) {

        case 1:

            s.sortdata();

            break;

        case 2:

            s.msort();

            break;

        default:

            cout << "Invalid option." << endl;

        }

        cout << "Do you want to continue? (y/n): ";

        cin >> type;

    }

    return 0;

}