Homework 4

5-1

language independent

step 1: understand the problem - data flow diagra

step 2: design a solution - { data structures
algorithm - psudo c

step 3: design test cases

step 4: implement the solution (i.e., program).

step 5: test the program.

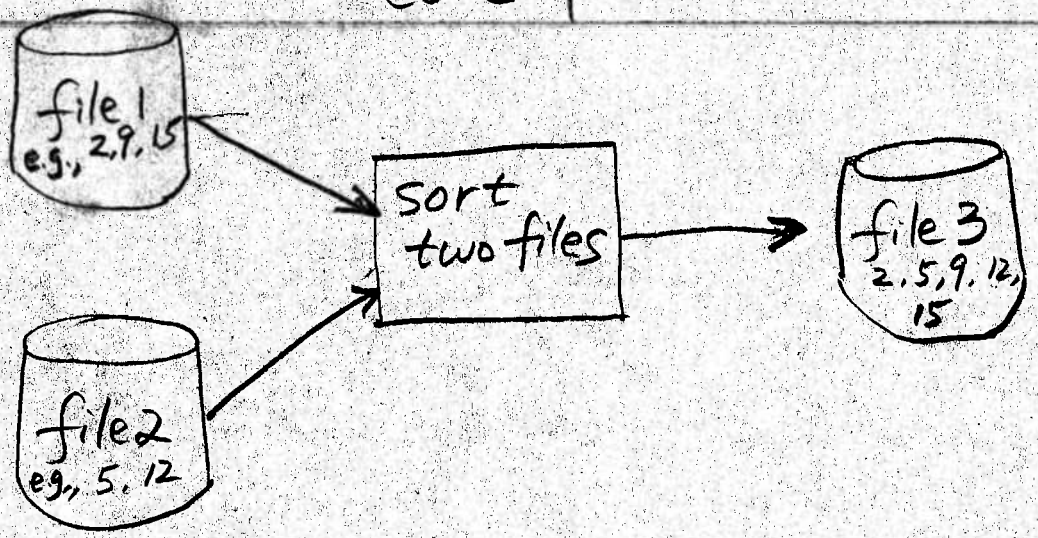
language dependent

Homework 4

steps 1~2 : 40min ~ 1h ~~5~~ = 2

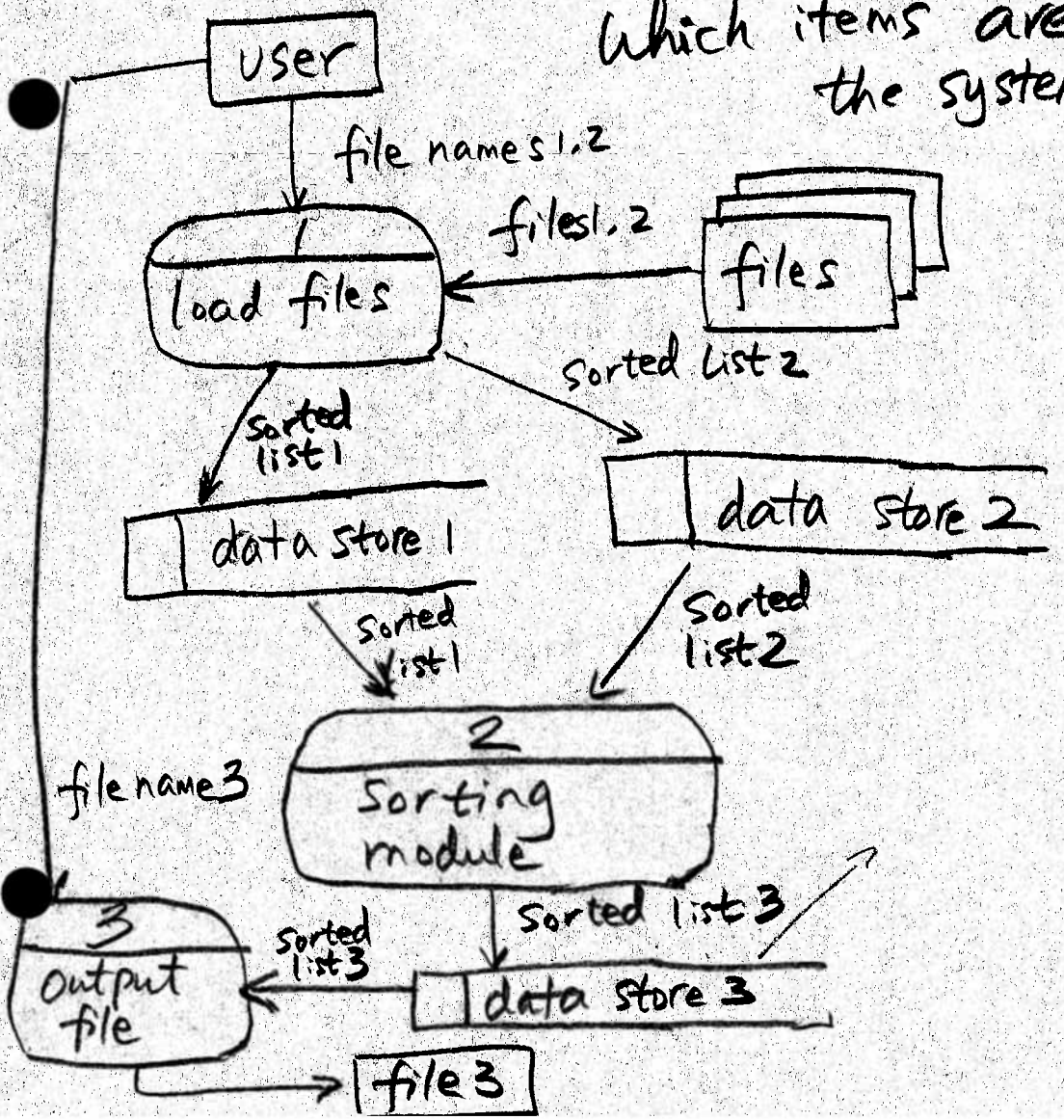
- step 1 :

core part



Which items are external to the system ?

Notation in DFD - Data Flow Diagram



step 2:

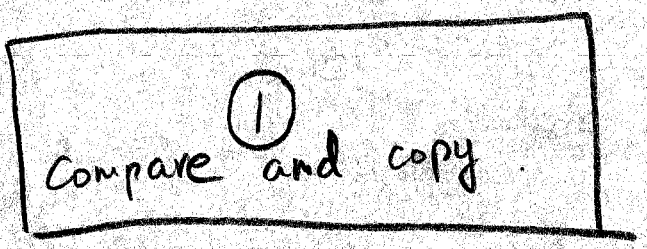
data structures: Sorted lists — arrays

algorithm:

See basic idea on page 4

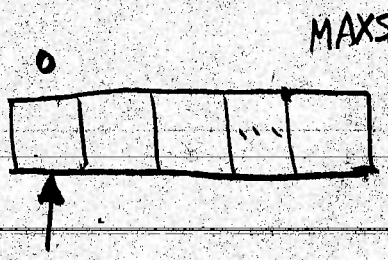
② repeat ① until finish one of the two input arrays. How ?

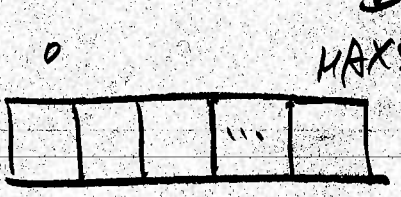
```
index1 ← 0;
index2 ← 0;
index3 ← 0;
while ( index1 < ary_size1  && index2 < ary_size2)
```

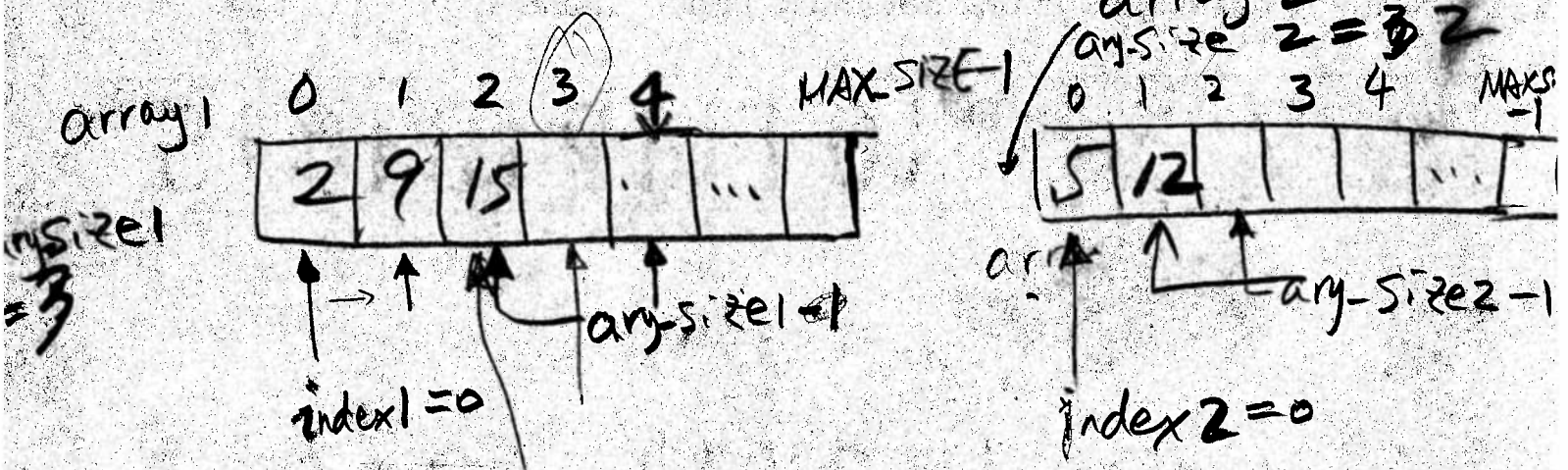


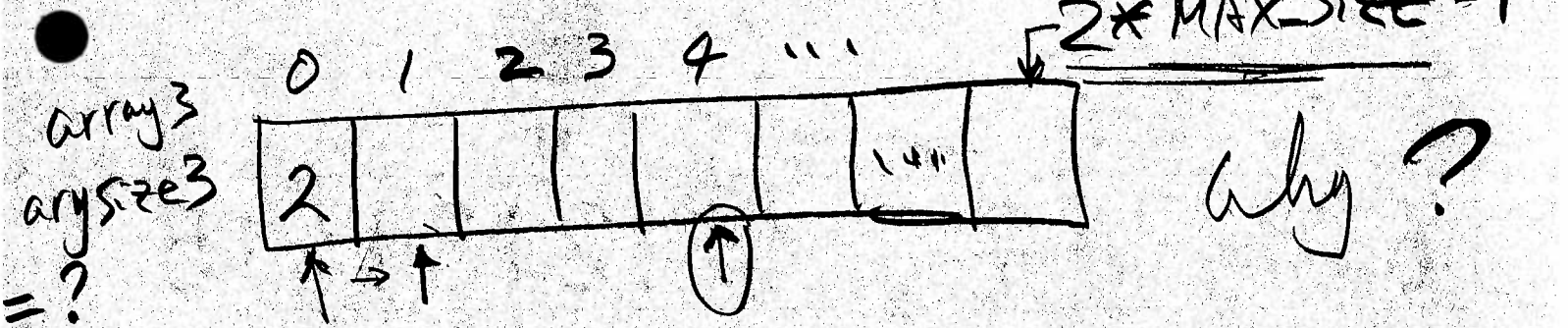

```
}
if (index1 == ary_size1) { /* copy rest of array 2 into array 3 */
    for( i= index2; i<ary_size2; i++){ How ?
        array3[index3] ← array2[index2];
        index3++; /* Do NOT forget this */
    }
} else { /* copy rest of array 1 into array 3 */
    for (i=index3; i<ary_size1; i++) {
        array3[index3] ← array1[index1]
        index3++;
    }
}
ary_size3 ← index3;
return ary_size3;
```

5-4









why ?

① compare array1[index1] and array2[index2]

if array1[index1] < array2[index2]

array3[index3] ← array1[index1];

index3++;

index1++;

} else {

array3[index3] ← array2[index2];

index2++;

Interface: Input/output of the algorithm:

Input: array 1, ary-size 1
int array 2, ary-size 2

Output: array 3, ary-size 3

Implementation: (prototype)

```
int sort_arrays(int array1[ ], int ary1_size1,
                int array2[ ], int ary_size2,
                int array3[ ]);
```

↑ ary-size3

```
void sort_arrays(int array1[], int ary-size1, int array2[],
     int ary-size2; int array3[], int& ary-size3,
```

↑ call by reference

data type: unsign int ⟶ int