

1. Write a program with below operations for a disjoint set forest. Elements will be of integer numbers from 0 to size-1.
 - a. **void** `init(DSF& forest, int size)` – which initialize the forest `f` with a size `size`.
 - b. **void** `makeOneElementSets(DSF& forest)` – call `makeSet()` function for all elements from 0 to size-1.
 - c. **int** `find(DSF& forest, int index)` – find and return the representant of a set with element `index`.
 - d. **void** `makeUnion(DSF& forest, int index1, int index2)` – make a union of a tree with element `index1` and a tree with element `index2`. **Assumption:** this function will be called only for elements from different trees.
 - e. **int** `parent(DSF& forest, int index)` – return the index of parent of element `index`.

For **10 points** present solutions for this list till **Week 11**.

For **8 points** present solutions for this list till **Week 12**.

For **5 points** present solutions for this list till **Week 13**.

After Week 13 the list is closed.

Appendix 1

The solution will be automated tested with tests from console of presented below format. The test assumes, that there are up to `X` different forests, which there are created as the first operation in the test. Each forest can be initialized separately.

If a line is empty or starts from ``#'` sign, the line have to be ignored.

In any other case, your program should print an exclamation mark and write (copy) introduced a line and then, depending on the command follow the correct procedure / function.

If a line has a format:

`GO X`

your program has to create `n` forests (without initialization). The forests are numbered from 0 like an array of forests. Default current forest is a forest with number 0. This operation will be called once as the first command.

If a line has a format:

`CH n`

your program has to choose a forest of a number `n`, and all next functions will operate on this tree. There is $n \geq 0$ and $n < X$.

If a line has a format:

`IN v`

your program has to call `init(f, v)` for current forest `f`. For any forest this operation will be called once, before using the tree.

If a line has a format:

MO

your program has to call `makeOneElementSets(f)` for current forest `f`.

If a line has a format:

FD `v`

your program has to call `find(f, v)` for current forest `f` and a node `v` and write a line with result value on the console.

UN `v u`

your program has to call `makeUnion(f, v, u)` for current forest `f`, and two nodes `v` and `u`.

If a line has a format:

PA `v`

your program has to call `parent(f, v)` for current forest `f` and write the answer on the console.

If a line has a format:

HA

your program has to end the execution, writing as the last line “END OF EXECUTION”.
Every test ends with this line.

For example for input test:

```
GO 2
IN 8
MO
PA 1
UN 1 2
PA 1
FD 1
HA
```

The output have to be:

```
START
!GO 2
!IN 8
!MO
!PA 1
1
!UN 1 2
!PA 1
2
!FD 1
2
!HA
```

END OF EXECUTION