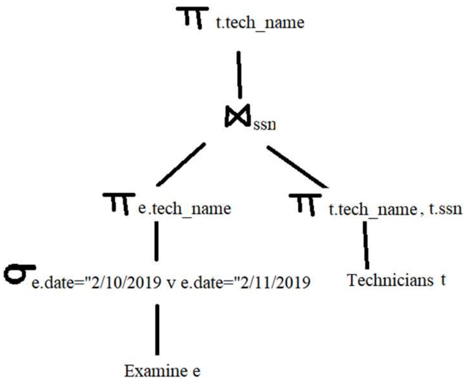
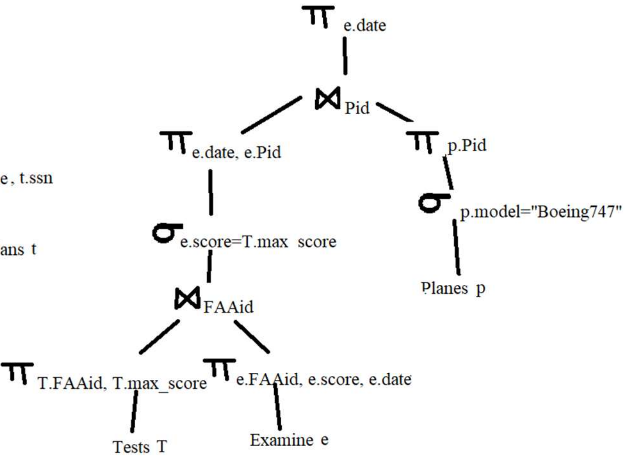
1)

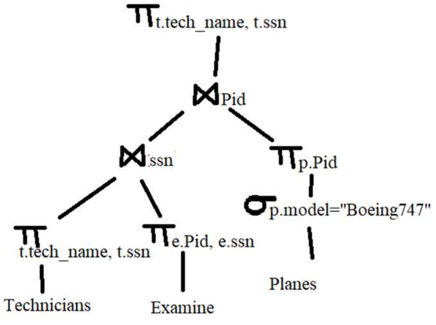
$\pi_{t.tech\_name}$
$\sigma_{e.date=\text{"}2/10/2019\text{ v }e.date=\text{"}2/11/2019}$
Technicians t $\bowtie_{ssn}$ Examine e



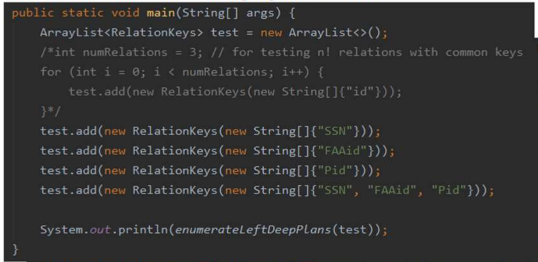
$\pi_{e.date}$
$\sigma_{e.score=T.max\_score\ \wedge\ p.model=\text{"Boeing747"}}$
Planes p $\bowtie_{Pid}$ Examine e $\bowtie_{FAAid}$ Tests T



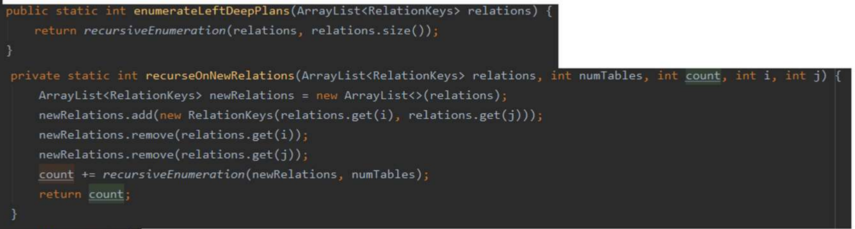
$\pi_{t.tech\_name,\ t.ssn}$
$\sigma_{p.model=\text{"Boeing747"}}$
Planes $\bowtie_{Pid}$ Examine $\bowtie_{ssn}$ Technicians



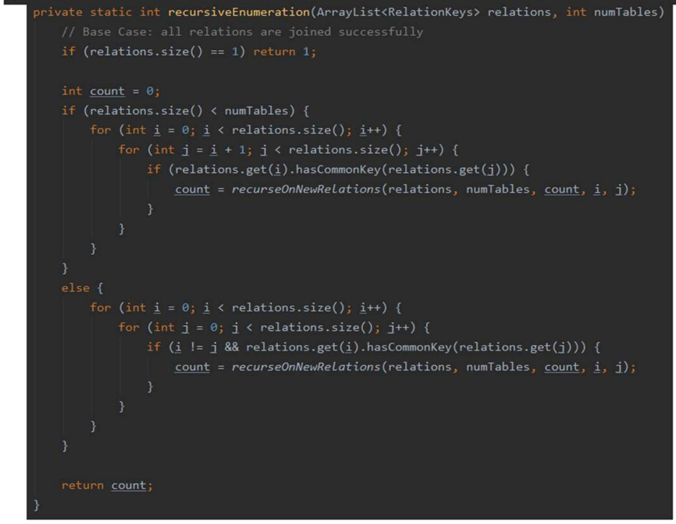
1c) 12. Calculated with a Java program that I wrote.



```java
public static void main(String[] args) {
    ArrayList<RelationKeys> test = new ArrayList<>();
    /*int numRelations = 3; // for testing n! relations with common keys
    for (int i = 0; i < numRelations; i++) {
        test.add(new RelationKeys(new String[]{"id"}));
    }*/
    test.add(new RelationKeys(new String[]{"SSN"}));
    test.add(new RelationKeys(new String[]{"FAAid"}));
    test.add(new RelationKeys(new String[]{"Pid"}));
    test.add(new RelationKeys(new String[]{"SSN", "FAAid", "Pid"}));

    System.out.println(enumerateLeftDeepPlans(test));
}
```

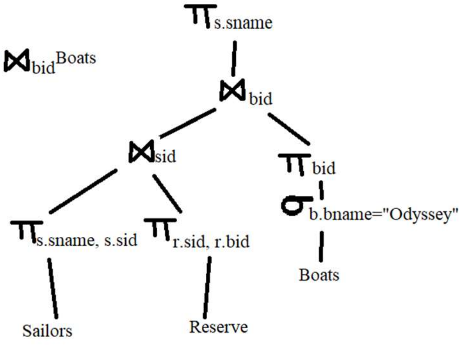


```java
public static int enumerateLeftDeepPlans(ArrayList<RelationKeys> relations) {
    return recursiveEnumeration(relations, relations.size());
}

private static int recurseOnNewRelations(ArrayList<RelationKeys> relations, int numTables, int count, int i, int j) {
    ArrayList<RelationKeys> newRelations = new ArrayList<>(relations);
    newRelations.add(new RelationKeys(relations.get(i), relations.get(j)));
    newRelations.remove(relations.get(i));
    newRelations.remove(relations.get(j));
    count += recursiveEnumeration(newRelations, numTables);
    return count;
}
```

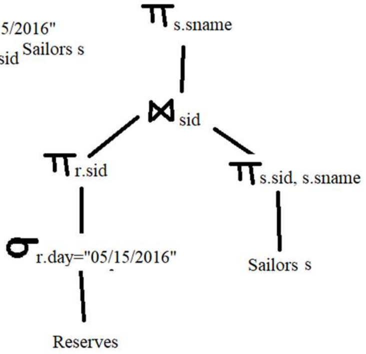


```java
private static int recursiveEnumeration(ArrayList<RelationKeys> relations, int numTables)
    // Base Case: all relations are joined successfully
    if (relations.size() == 1) return 1;

    int count = 0;
    if (relations.size() < numTables) {
        for (int i = 0; i < relations.size(); i++) {
            for (int j = i + 1; j < relations.size(); j++) {
                if (relations.get(i).hasCommonKey(relations.get(j))) {
                    count = recurseOnNewRelations(relations, numTables, count, i, j);
                }
            }
        }
    }
    else {
        for (int i = 0; i < relations.size(); i++) {
            for (int j = 0; j < relations.size(); j++) {
                if (i != j && relations.get(i).hasCommonKey(relations.get(j))) {
                    count = recurseOnNewRelations(relations, numTables, count, i, j);
                }
            }
        }
    }

    return count;
}
```

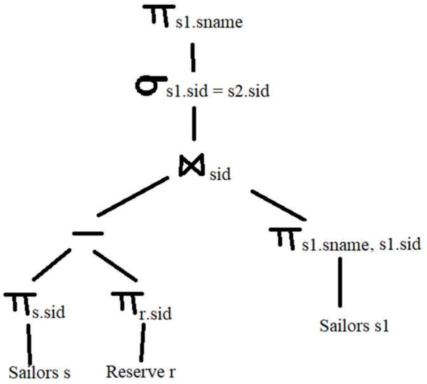
2) $\pi_{s.sname}$
$\sigma_{b.bname=\text{"Odyssey"}}$
Sailors s $\bowtie_{sid}$ Reserve $\bowtie_{bid}$ Boats



$\pi_{s.sname}$
$\sigma_{r.day=\text{"}05/15/2016\text{"}}$
Reserve r $\bowtie_{sid}$ Sailors s



$\pi_{s1.sname}$
$\sigma_{s1.sid\ =\ s2.sid}$
$\left(\pi_{s.sid}\text{Sailors s} - \pi_{r.sid}\text{Reserve}\right)$ s2 $\bowtie_{sid}$ Sailors s1


π s1.sname
σ s1.sid = s2.sid
⋈ sid
—
π s1.sname, s1.sid
π s.sid
π r.sid
Sailors s1
Sailors s
Reserve r