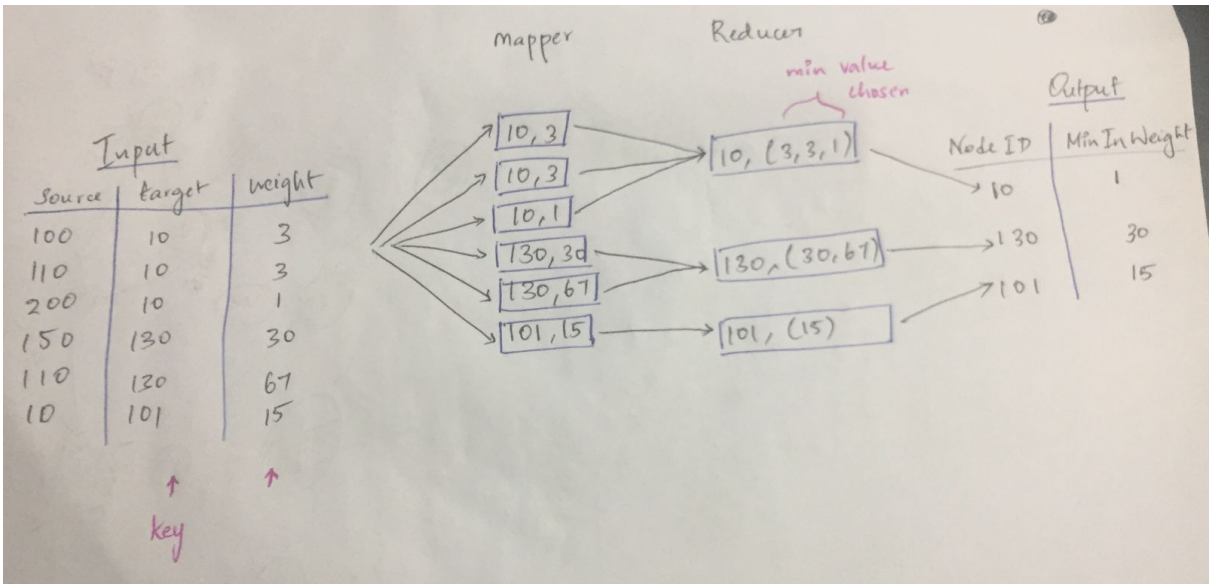
1)a




**EXPLANATION**: The input has source ,target and weights. Since we are only supposed to find the smallest weight among each node's inbound edges, we take only the target nodes and its respective weights. In the mapper phase, each target and weights are mapped making the target node the key. In the combiner phase, the weights of unique key are joined together. In the reducer phase, the minimum weight amongst all the weights combined for a given target node is chosen and outputted with its respective key (node).

1)b

**PSEUDO CODE**

**Mapper1():**

```
Parse each line in input file as line
Split '\t' and store each line in string array parts
If part[0] = Student
        CompositeKey=(part[2],1)
        Value = part[1]
Else
        CompositeKey=(part[1],2)
        Value = part[2]
```

**Reducer1():**

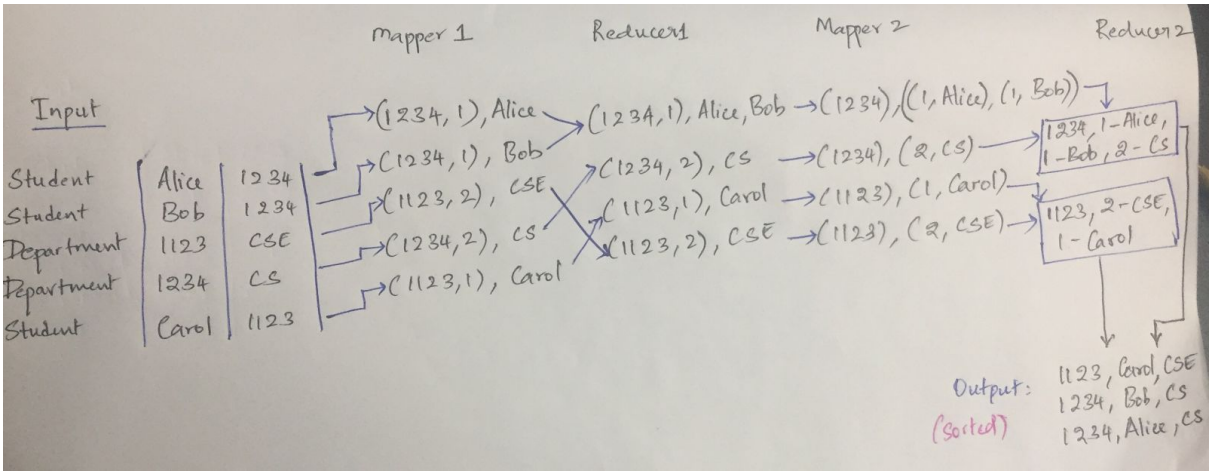
```
Pass the joined values to Mapper2()
```

**Mapper2():**

```
newKey = CompositeKey[0]
For all the values k
        newValue=[CompositeKey[1],Value[k]]
```

**Reducer2()**:

```
Based on the Combiner's output, find Department using Value=2
For all values k, context.write(key,k,dept)
```




**EXPLANATION**:

- **Mapper1()** parses each line in input file. It converts every line into a composite key of JoinKey and Tag. Join key is the DepartmentID and Tag indicates whether it belongs to Student or Department using values 1 and 2 **(1 for Student and 2 for Department)** . Value are the respective values based on the Tag.
- **Reducer1()** is used to club all the values corresponding to the composite key.
- **Mapper2()** Retrieves JoinKey from TaggedKey.
- **Reducer2()** uses the newKey (JoinKey) to combine the corresponding values.
- We sort the output based on ascending order of DepartmentID