BIG DATA MANAGEMENT

CZ/CE4123

Tutorial 9
MapReduce Designs

QUESTION1

```
Map(string key, string value){
    if (key.equals("Student-Table")){
        studentID=split(value).first;
        courseID=split(value).second;
        Emit(studentID, courseID);
    }
}
```

QUESTION1

```
Reduce(string key, iterator values){
    int s=0;
    Map<string, string> distinct_course;
    for(each v in values){
        if(distinct_course does not contain v)
         {
            distinct_course.insert<v,1>;
            s++;

         }

    }
    Emit(key, s);
}
```

QUESTION 2

```
Map(string key, string value){
    if(key.equals("Student-Table")){
        studentID=split(value).first;
        courseID=split(value).second;
        semester=split(value).third;
        Emit(courseID, semester);
    }
}
```

QUESTION 2

```
Reduce(string key, iterator semesters){
    Map<string, string> semester_freq;
    for(each sem in semesters){
      if(semester_freq does not contain sem)
      {
            semester_freq.insert<sem, 1>;
      }
      else
            semester_freq[sem]++;
    }
    int cnt=0;

    for(each <semester, freq> in semester_freq){
        if(freq>50){
            cnt++;
        }
    }
    if (cnt>=2)
        Emit(courseID, NULL);
}
```

QUESTION 3

```
Map1(string key, string value){
    if(key.equals("Student-Table")){
        studentID=split(value).first;
        courseID=split(value).second;
        semester=split(value).third;
        Emit(toString(courseID, ";", semester), toString("s;", studentID));
    }
    if(key.equals("Professor-Table")){
        professorID=split(value).first;
        courseID=split(value).second;
        semester=split(value).third;
        Emit(toString(courseID, ";", semester), toString("p;", professorID));
    }
}
```

QUESTION 3

```
Reduce1(string key, iterator values){
    List professors;
    List students;
    for(each value in values){
        if(value starts with "s"){
            students.add(getStudentID(value));
        }
        if(value starts with "p"){
            professors.add(getProfessorID(value));
        }
    }
    for(each student in students){
        for(each professor in professors){
            Emit (student, professor);
        }
}
```

QUESTION 3

```
//Map2 takes the output of Reduce1, with the purpose to remove duplicates.

Map2(string student, string professor){

    Emit(toString(student,”;”, professor), “1”);

}

Reduce2(string key, iterator values){

    student=split(key).first;

    professor=split(key).second;

    Emit(student, professor);

}
```