In [ ]:
```
'Week 5 Practice Programming Assignment'

# Information about courses:  Course Code~Course Name~Semester~Year~Instructor
# Information about students: Roll Number~Full Name
# Information about grades:   Course Code~Semester~Year~Roll Number~Grade

'''Grade is a Relation which can be joined to both Courses, Students
In this problem we don't need to do join on Courses although'''

# Function body for testing from here, with parameter 'strm'

g2m = {'A': 10, 'AB': 9, 'B': 8, 'BC': 7, 'C': 6, 'CD': 5, 'D': 4}

Courses  = {} # Course Code can be used as key, will not be used anywhere
Students = {} # Roll Number can be used as key

# Roll Number used as keyfor problem, while actually no key possible
Grades   = {} # Just Roll Number to Grade_list mapping

# st = [ln.strip() for ln in strm.strip().split('\n')] # if placed in function for testing
state = None
while True: #for ln in st: # if placed in function for testing
    ln = input().strip() # Comment this if testing as function
    if ln=='EndOfInput':
        break
    elif ln in ('Courses','Grades','Students'):
        state = ln
        continue
    elif state=='Courses':
        ln = ln.split('~')
        Courses[ln[0]]  = ln[1:]
    elif state=='Students':
        ln = ln.split('~')
        Students[ln[0]] = ln[1]
    elif state=='Grades':
        ln = ln.split('~')
        if ln[3] not in Courses:
            Courses[ln[3]] = [ ln[4] ]
        else:
            Courses[ln[3]].append( ln[4] )
    else:
        pass

for rn in sorted(Students):
    if rn in Courses:
        avg = round( sum(g2m[x] for x in Courses[rn]) / len(Courses[rn]) , 2 )
    else:
        avg = 0
    print( '~'.join( (rn, Students[rn], str(avg)) ) )
```

In [ ]:
```
'Testing Part of Above Function'
# for i in TC:
#     f(TC[i])
#     print('\n'*5)
```

In [ ]:
```
```

In [ ]:
```
```

In [ ]:

```
'Week 5 Programming Assignment'

# Information about books: Accession Number~Title
# Information about borrowers: Username~Full Name
# Information about checkouts: Username~Accession Number~Due Date
# Note: Due Date is in YYYY-MM-DD format.

'''Acc Number as key in Books
Username as key in Borrowers
Due date as key in Checkouts due to nature of problem
'''

Books  = {} # Accession Number can be used as key
Borrowers = {} # User Name can be used as key
Checkouts   = {} # Checkout dates will be used as key

state = None
while True:
    ln = input().strip() # Comment this if testing as function
    if ln=='EndOfInput':
        break
    elif ln in ('Books','Borrowers','Checkouts'):
        state = ln
        continue
    elif state=='Books':
        ln = ln.split('~')
        Books[ln[0]]  = ln[1]
    elif state=='Borrowers':
        ln = ln.split('~')
        Borrowers[ln[0]] = ln[1]
    elif state=='Checkouts':
        ln = ln.split('~')
        if ln[2] not in Checkouts:
            Checkouts[ln[2]] = [ ln[:2] ]
        else:
            Checkouts[ln[2]].append( ln[:2] )
    else:
        pass

res = []
for date in Checkouts:
    entries = Checkouts[date]
    for entry in entries:
        full_name = Borrowers[ entry[0] ]
        book_name = Books[ entry[1] ]
        res.append( '~'.join((date,full_name,entry[1],book_name)) )

for entry in sorted(res):
    print( entry )
```