IEMS HOMEWORK1

a. i) Map/Reduce codes

mapper_a.py:

```python
#!/usr/bin/env python

import sys

# input comes from STDIN (standard input)
for line in sys.stdin   :
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    ratingPair = list(line.split())
    # # increase counters
    usrID = ratingPair[0]
    movieID = ratingPair[1]
    rating = ratingPair[2]

    print('%s\t%s\t%s' % (usrID, movieID, rating))
```

Reducer_a.py:

```python
#!/usr/bin/env python
"""reducer.py"""

from operator import itemgetter
import sys

current_userID = None
current_movieID = None
current_count = 0
count = 0
current_rating = None
rating = None
userID = None
movieID = None



# input comes from STDIN
```

```
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # parse the input we got from mapper.py
    userID, movieID, rating, count = line.split('\t',1)
    #put userID into list
    userID = list[]
    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:

        continue

    # this IF-switch only works because Hadoop sorts map output
    # by key (here: word) before it is passed to the reducer

    #combine the userID in pairs
    def combine(userID, n):
        answers = []
        one = [0] * n
        def next_c(li = 0, ni = 0):
        if ni == n:
            answers.append(copy.copy(one))
            return
        for lj in xrange(li, len(userID)):
            one[ni] = userID[lj]
            next_c(lj + 1, ni + 1)
        next_c()
        return answers

    #when two ratings are the same, count+1
    if current_rating == rating:
        if current_movieID == movieID:
        current_count += count
    else:
        if current_rating:
            # write result to STDOUT
            print '%s\t%s' % (current_rating, current_count)
        current_count = count

# do not forget to output the last word if needed!
    sorted(count)
    print '%s\t%s\t%s'(userID, movieID, rating)
```

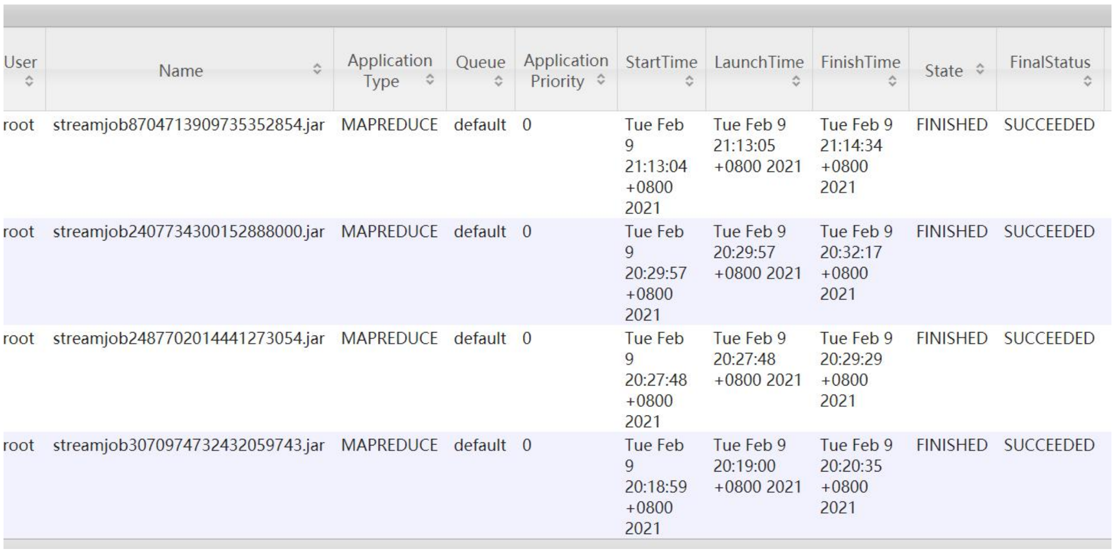
b. mapper_b.py:

```
#!/usr/bin/env python

import sys

# input comes from STDIN (standard input)
for line in sys.stdin   :
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    ratingPair = list(line.split())
    # # increase counters
    usrID = ratingPair[0]
    movieID = ratingPair[1]
    rating = ratingPair[2]
    # Switch the output format and make userID the key
print('%s\t%s\t%s' % (usrID, movieID, rating))
```

c.



User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus
root	streamjob8704713909735352854.jar	MAPREDUCE	default	0	Tue Feb 9 21:13:04 +0800 2021	Tue Feb 9 21:13:05 +0800 2021	Tue Feb 9 21:14:34 +0800 2021	FINISHED	SUCCEEDED
root	streamjob2407734300152888000.jar	MAPREDUCE	default	0	Tue Feb 9 20:29:57 +0800 2021	Tue Feb 9 20:29:57 +0800 2021	Tue Feb 9 20:32:17 +0800 2021	FINISHED	SUCCEEDED
root	streamjob2487702014441273054.jar	MAPREDUCE	default	0	Tue Feb 9 20:27:48 +0800 2021	Tue Feb 9 20:27:48 +0800 2021	Tue Feb 9 20:29:29 +0800 2021	FINISHED	SUCCEEDED
root	streamjob3070974732432059743.jar	MAPREDUCE	default	0	Tue Feb 9 20:18:59 +0800 2021	Tue Feb 9 20:19:00 +0800 2021	Tue Feb 9 20:20:35 +0800 2021	FINISHED	SUCCEEDED

Maximum mapper time	Minimum mapper time	Average mapper time	Maximum reducer time	Minimum reducer time	Average reducer time	Total job
45s	30s	36.25s	95s	60s	70.5s	7m7s