Project 3 - CMPE Car Trip
CmpE 250, Data Structures and Algorithms, Fall 2023

Instructor: Bahri Atay Özgövde
TAs: Suzan Ece Ada, Kutay Altıntaş
SAs: Işıl Su Karakuzu, Atakan Yaşar

Due: December 10, 2023, 23:59 (Strict)

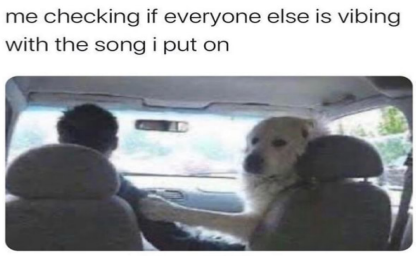
1 Introduction

Alright, the CMPE car trip is about to start, and everything's set, including the rented 1987 Ford F-250. After some chill debates, you claimed the front seat and got the title of the side seat princess. You throw everyone in the pickup bed. But now, you have got a job. You are in charge of the playlist. Knowing it could be tricky, you decide not to leave it to chance. With insights from CMPE250, you are all set to create the ultimate playlist.

Your game plan involves low-key stalking the Spotify accounts of everyone on the trip to grab their playlists, with the main goal being to figure out a way to cook up the ultimate collaborative playlist, the **EpicBlend**, using all this info.

It is all about keeping a consistent vibe, starting from the most jammed tunes down to the least, making sure the excitement stays high throughout the trip. However, keep in mind that this journey might take longer than expected. Throughout this process, you must monitor your friends' modifications in their playlists and the play count of the songs you have collected. Get ready for a musical journey like no other!

(a) 1987 Ford F-250



(b) The chaotic situation

2 Details

You have gathered N playlists (provided as input). Each playlist has its own set of unique songs. Moreover, no two playlists share the same songs; each one is distinct.

Every song within each playlist is categorized into three vibe-worthy categories: *Heartache*, *Roadtrip*, and *Blissful*. A song is assigned three scores based on its relevance to the listed categories, ranging between $[0, 100]$.

Each song has a play count, which remains unchanged throughout crafting EpicBlend.

The ultimate playlist you are creating from your friends' playlists, EpicBlend, should include at most a specified number of songs from each category, denoted as c_h, c_r, and c_b.

EpicBlend should include songs with the highest category scores among the songs in the playlists you collected from your friends' accounts. In the event of a tie, songs with lexicographically smaller names should be prioritized. Remember, **EpicBlend cannot contain duplicate songs**; each song must be distinct.

However, a song **might be included under two or three different categories** in the playlist at the same time **if it achieves a sufficient score in both**.

Additionally, it is important to bear in mind that there is a limit, denoted as l, for the number of songs that can be added to each category in every playlist. For example, if the limit is set at 10, each playlist can individually add up to 10 songs in each category.

Once again, if the same song can enter EpicBlend under two categories, it will **concurrently decrease** the remaining capacity for **both** of these categories in the playlist.

Here is what might change in your friends' playlists:

1. A new song j can be added to playlist i.
2. A song j can be removed from its playlist.

These events may cause a change in the EpicBlend. Some new songs will enter and some songs will leave the EpicBlend, after the changes in your friends' playlists. So, your structure must handle these events in an efficient way.

Finally, you provided your friends with five chances to request information about EpicBlend. When your friends ask, you'll share all EpicBlend songs organized by their play count in descending order. In case of tie, songs will be listed lexicographically by their names.

3 Input & Output

3.1 Input

- There will be 2 input files named <song> and <test-case>. A single song file will be utilized for more than one test case.

3.1.1 Song File

- This file contains a database of songs you will use for various cases.
- The first line contains the number of songs, denoted as S.
- The following S lines contain songs, presenting their song ID, play count, and scores for the three categories, all organized in sequential order.
- The structure of the songs is as follows:

  <song-id><song-name><play-count><heartache-score><roadtrip-score><blissful-score>

  Example: 1 Song1 100 100 0 50

  Explanation: Song1 has an ID of 1, with a play count 100. Its Heartache score is 100, Roadtrip score is 0, and Blissful score is 50.

- It is **not mandatory** for all songs in the song file to be present in a playlist. There can be songs left out.

Song File
5
1 song1 100 99 0 50
2 song2 200 80 10 20
3 song3 50 70 20 30
4 song4 30 10 60 70
5 song5 10 50 50 50

Table 1: Sample Song File

3.1.2 Test Case File

- This file contains the case that you are working on.
- The first line includes four integers: l, indicating the category addition limit for each playlist, and c_h, c_r, and c_b, representing the song counts for the EpicBlend you are

crafting based on each of the three categories.

<playlist-category-limit> <heartache-category-limit><roadtrip-category-limit><blissful-category-limit>

Example: 4 3 8 10

Explanation: Each playlist can contribute EpicBlend a maximum of 4 songs to each category, meaning you can add up to 4 Heartache, 4 Roadtrip, and 4 Blissful songs from each playlist. Moreover, EpicBlend can contain a maximum of 3 Heartache songs, 8 Roadtrip songs, and 10 Blissful songs.

- The second line contains the number of playlists, denoted as N.
- The subsequent $2N$ lines consist of pairs. Each pair starts with two integers: the playlist ID and the number of songs in that playlist. The second line contains the song IDs represented by integers, and their count is the number of songs in that playlist. The properties of these songs can be obtained from the song database file.

  <playlist-id><playlist-song-count>
  <song-1><song-2>....<song-playlist-song-count>

  Example: 2 5
  4 6 78 65 100

  Explanation: The playlist with ID 2 has 5 songs at the beginning, consisting of songs with IDs 4, 6, 78, 65, and 100.

- After this part, the operations come.
- The following line contains the number of events, denoted as E.
- The next E lines contain one of the three event types: ADD, REM, or ASK.

  ADDITION EVENT:

  ADD <song-id> <playlist-id>

  One of your friends adds the song with ID <song-id> to Playlist ID <playlist-id>.

  You should add the chosen song to the playlist, which might change EpicBlend, too. For instance, if the added song's Heartache score is higher than the score of the previously added song in EpicBlend's Heartache category, the existing song in EpicBlend should be replaced with the newly added song in the playlist.

Note: It is guaranteed that the song is not currently on any playlists.

REMOVAL EVENT:

REM <song-id> <playlist-id>

One of your friends removes the song with ID <song-id> from Playlist ID <playlist-id>.

You should remove the selected song from the playlist, which might also impact EpicBlend. For example, if the removed song is already in EpicBlend in a category, it should be replaced with the next highest-scored song from the entire set of playlists that have not reached the limit l. Notice that it is possible that the removed song might be replaced by a song that is counted for another category in EpicBlend.

Note: The song is guaranteed to be in the given playlist.

ASKING EVENT:

ASK

One of your friends asks you to print the EpicBlend.

You should print EpicBlend in descending order of play counts. In the case of tied counts, the song with lexicographically smaller names is printed first.

Note: It is given that the ASK event will not occur more than 5 times.

Test Case File	Explanation
1 2 2 2	Each playlist you have gathered can contribute at most 1 song to Epic Blend in each category. EpicBlend can include up to 2 Heartache songs, 2 Roadtrip songs, and 2 Blissful songs in total.
2	There are 2 playlists.
1 2	Playlist with ID 1 contains 2 songs.
1 2	Playlist with ID 1 contains songs with IDs 1 and 2.
2 3	Playlist with ID 2 contains 3 songs.
3 4 5	Playlist with ID 2 contains songs with IDs 3, 4 and, 5.
3	There will be 3 events.
REM 3 2	One of your friends removes the song with ID 3 from Playlist ID 2.
ADD 3 1	One of your friends adds the song with ID 3 to Playlist ID 1.
ASK	One of your friends asks you to print the EpicBlend.

Table 2: Sample Test Case File with Explanations

3.2 Output

- At the beginning, you must create EpicBlend using the playlists you've collected from your friends. During this step, there is no need to print anything.
- After each ADD or REMOVE event, print the changes in EpicBlend across three categories. Output 2 lines, each containing 3 integers. The first line should represent the ID of added songs for each category, and the second line should represent the ID of removed songs for each category. Remember that the addition and removal in EpicBlend are based on score differences, and EpicBlend always includes the highest-scored possible songs in each category.

  <added-heartache><added-roadtrip><added-blissful>
  <removed-heartache><removed-roadtrip><removed-blissful>

  Example: 0 0 1
  0 0 2

  Explanation: The song with ID 2 is removed from Blissful category of EpicBlend because a higher-scored song with ID 1 now exists in the Blissful category. Consequently, the song with ID 1 is added to EpicBlend. No changes occurred in the

Heartache and Roadtrip categories.

- User input **does not involve** the addition or removal of songs directly to the EpicBlend. **Do not forget, EpicBlend changes are the events' consequences.**
- After each ASK event, you should print the song IDs of the songs in EpicBlend, following the given rules for order, in a single line with IDs separated by spaces.

So the examination and final output of the given input is:

At the beginning, there are two playlists with IDs 1 and 2.

Playlist 1
1 song1 100 99 0 50
2 song2 200 80 10 20

Playlist 2
3 song3 50 70 20 30
4 song4 30 10 60 70
5 song5 10 50 50 50

	Heartache	Roadtrip	Blissful
Added to EpicBlend	song1(99) song3(70)	song4(60) song2(10)	song4(70) song1(50)
Added From Playlist 1:	1	1	1
Added From Playlist 2:	1	1	1
Not Added Songs	song2(80) song5(50) song4(10)	song5(50) song3(20) song1(0)	song5(50) song3(30) song2(20)

Firstly, let us examine the Heartache category. We start with the highest-scored song, Song 1 from Playlist 1. We include Song 1 in the EpicBlend. EpicBlend allows up to 2 Heartache songs, so we check the second-highest scoring song, Song 2. However, there's a limit on playlists: in this case, only 1 song from each playlist can be added under each category. Since Song 2 is in the same playlist as Song 1, we skipped it and chose the third-ranking song, Song 3. Then, we move on to the next category.

Now we focus on the Roadtrip category, beginning with the top-scoring song, Song 4 from Playlist 2. We include Song 4 in the EpicBlend. EpicBlend permits the inclusion of up to 2 songs in the Roadtrip category. Moving on to assess the second-highest scoring song, namely Song 5, we encounter a limitation on playlists—only 1 song from each category can be added per playlist. As Song 5 shares a playlist with Song 4, we bypass it and move to the third-ranking song, Song 3. However, Song 3, too, shares a playlist with Song 4. So, we choose the fourth-best song, Song 2, because it is from a different playlist, Playlist 1. Before adding Song 2 to the Roadtrip category, we ensure that Playlist 1 has not surpassed its limit for this category, and upon confirmation, we

proceed with the inclusion of Song 2.

Lastly, we explore the Blissful category. Beginning with the top-scoring song, Song 4 from Playlist 2, we add it to EpicBlend. Following that, we consider the second-highest scoring song, Song 1 from Playlist 1. After confirming that Playlist 1 and Playlist 2 haven't exceeded their category limits, we include these songs in EpicBlend.

An important note: observe that Song 4 and Song 1 are included in EpicBlend under two categories, yet they are listed only once. EpicBlend avoids duplicate songs but permits songs to be included from multiple categories.

The songs in EpicBlend are arranged based on their play count, with lexicographic order used as a tiebreaker for songs with the same play count.

Taking these into account, EpicBlend looks like this:

EpicBlend
2 song2 200 80 10 20
1 song1 100 99 0 50
3 song3 50 70 20 30
4 song4 30 10 60 70

Event 1: REM 3 2

Playlist 1
1 song1 100 99 0 50
2 song2 200 80 10 20

Playlist 2
~~3 song3 50 70 20 30~~
4 song4 30 10 60 70
5 song5 10 50 50 50

	Heartache	Roadtrip	Blissful
Added to EpicBlend	song1(99) ~~song3(70)~~ song5(50)	song4(60) song2(10)	song4(70) song1(50)
Added From Playlist 1:	1	1	1
Added From Playlist 2:	1	1	1
Not Added Songs	song2(80) song4(10)	song5(50) ~~song3(20)~~ song1(0)	song5(50) ~~song3(30)~~ song2(20)

In this event, Song 3 is removed from Playlist 2. In the Heartache category, Song 3 is removed. So, we must replace it with the highest-scoring suitable song, Song 5. We

ensure that neither playlist exceeds its limit. The other categories are not affected by this event.

Output for REM 3 2
5 0 0
3 0 0

As can be seen in the previous table, Song 5 entered the Heartache category, and Song 3 was removed from the Heartache category. And there is no change in other categories.

EpicBlend
2 song2 200 80 10 20
1 song1 100 99 0 50
~~3 song3 50 70 20 30~~
4 song4 30 10 60 70
5 song5 10 50 50 50

Event 2: ADD 3 1

Playlist 1
1 song1 100 99 0 50
2 song2 200 80 10 20
3 song3 50 70 20 30

Playlist 2
4 song4 30 10 60 70
5 song5 10 50 50 50

	Heartache	Roadtrip	Blissful
Added to EpicBlend	song1(99) song5(50)	song4(60) song3(20) ~~song2(10)~~	song4(70) song1(50)
Added From Playlist 1:	1	1	1
Added From Playlist 2:	1	1	1
Not Added Songs	song2(80) song3(70) song4(10)	song5(50) song2(10) song1(0)	song5(50) song3(30) song2(20)

In this event, Song 3 is added to Playlist 1. Let us examine Roadtrip category. Song 3 has a higher score than Song 2. Thus, we should replace Song 2 with Song 3. We can do this operation because both are now on the same list, and no playlist exceeds their limit. In the Heartache category, Song 3 cannot replace Song 5 because Song 1 scores better in the same playlist.

Output for ADD 3 1
0 3 0
0 2 0

Song 3 entered the Roadtrip category and Song 2 was removed from the Roadtrip category. And there is no change in other categories.

EpicBlend
~~2 song2 200 80 10 20~~
1 song1 100 99 0 50
3 song3 50 70 20 30
4 song4 30 10 60 70
5 song5 10 50 50 50

Event 3: ASK

Output for ASK
1 3 4 5

Output is listed from the most played song to the least played song.

Output File
5 0 0
3 0 0
0 3 0
0 2 0
1 3 4 5

Table 3: Sample 1 Output File

3.3 More Examples

Song File
12
1 song1 100 99 0 50
2 song2 200 80 10 20
3 song3 50 70 20 30
4 song4 30 10 60 70
5 song5 10 50 50 50
6 song6 5 10 70 70
7 song7 44 90 0 100
8 song8 35 0 100 0
9 song9 4 0 20 60
10 song10 100 80 0 90
11 song11 300 0 80 80
12 song12 10000 100 100 100

Test Case File
2 1 3 6
3
1 2
3 5
2 1
4
3 1
1
10
ADD 9 2
ASK
ADD 10 2
ASK
ADD 11 2
ASK
ADD 12 2
ASK
REM 5 1
ASK

Table 4: Sample 2 Input Files

Output File
0 0 9
0 0 0
1 3 4 5 9
0 0 10
0 0 9
1 10 3 4 5
0 11 11
0 3 4
11 1 10 3 4 5
12 12 12
1 4 11
12 11 1 10 3 5
0 3 0
0 5 5
12 11 1 10 3

Table 5: Sample 2 Output File

At the beginning, there are three playlists. The first contains 3 and 5, the second contains 4, and the third contains 1. There is a limit 1 for Heartache, 3 for Roadtrip, 6 for Blissful. Each playlist can give at most 2 songs to EpicBlend in each category.

Before Events:

Playlist 1	Playlist 2	Playlist 3
song3 50 70 20 30 song5 10 50 50 50	song4 30 10 60 70	song1 100 99 0 50

	Heartache	Roadtrip	Blissful
	song1(99)	song4(60) song5(50) song3(20)	song4(70) song1(50) song5(50) song3(30)
From Playlist 1:	0	2	2
From Playlist 2:	0	1	1
From Playlist 3:	1	0	1

EpicBlend:	song1(100) song3(50) song4(30) song5(10)

3.3.1 Event 1: ADD 9 2

Playlist 1	Playlist 2	Playlist 3
song3 50 70 20 30 song5 10 50 50 50	song4 30 10 60 70 song9 4 0 20 60	song1 100 99 0 50

	Heartache	Roadtrip	Blissful
	song1(99)	song4(60) song5(50) song3(20)	song4(70) song9(60) song1(50) song5(50) song3(30)
From Playlist 1:	0	2	2
From Playlist 2:	0	1	2
From Playlist 3:	1	0	1

EpicBlend:	song1(100) song3(50) song4(30) song5(10) song9(4)

3.3.2 Event 2: ASK

EpicBlend:	song1(100) song3(50) song4(30) song5(10) song9(4)

3.3.3 Event 3: ADD 10 2

Playlist 1	Playlist 2	Playlist 3
song3 50 70 20 30 song5 10 50 50 50	song4 30 10 60 70 song9 4 0 20 60 song10 100 80 0 90	song1 100 99 0 50

	Heartache	Roadtrip	Blissful
	song1(99)	song4(60) song5(50) song3(20)	song10(90) song4(70) ~~song9(60)~~ song1(50) song5(50) song3(30)
From Playlist 1:	0	2	2
From Playlist 2:	0	1	2
From Playlist 3:	1	0	1

Notice that Playlist 2 **can't** give more than two songs in the Blissful category. So, we removed Song 9.

EpicBlend:	song1(100) song10(100) song3(50) song4(30) song5(10) ~~song9(4)~~

3.3.4 Event 4: ASK

EpicBlend:	song1(100) song10(100) song3(50) song4(30) song5(10)

3.3.5 Event 5: ADD 11 2

Playlist 1	Playlist 2	Playlist 3
song3 50 70 20 30 song5 10 50 50 50	song4 30 10 60 70 song9 4 0 20 60 song10 100 80 0 90 song11 300 0 80 80	song1 100 99 0 50

	Heartache	Roadtrip	Blissful
	song1(99)	song11(80) song4(60) song5(50) ~~song3(20)~~	song10(90) song11(80) ~~song4(70)~~ song1(50) song5(50) song3(30)
From Playlist 1:	0	1	2
From Playlist 2:	0	2	2
From Playlist 3:	1	0	1

Notice that Playlist 2 **cannot** give more than two songs in the Blissful category. So, we removed Song 4. For the category Roadtrip, at most 3 songs can participate. Therefore, we removed worst scored song in the Roadtrip category.

EpicBlend:	song11(300) song1(100) song10(100) song3(50) song4(30) song5(10)

3.3.6 Event 6: ASK

EpicBlend:	song11(300) song1(100) song10(100) song3(50) song4(30) song5(10)

3.3.7 Event 7: ADD 12 2

Playlist 1	Playlist 2	Playlist 3
song3 50 70 20 30 song5 10 50 50 50	song4 30 10 60 70 song9 4 0 20 60 song10 100 80 0 90 song11 300 0 80 80 song12 10000 100 100 100	song1 100 99 0 50

	Heartache	Roadtrip	Blissful
	song12(100) ~~song1(99)~~	song12(100) song11(80) ~~song4(60)~~ song5(50)	song12(100) song10(90) ~~song11(80)~~ song1(50) song5(50) song3(30)
From Playlist 1:	0	1	2
From Playlist 2:	1	2	2
From Playlist 3:	0	0	1

EpicBlend:	song12(10000) song11(300) song1(100) song10(100) song3(50) ~~song4(30)~~ song5(10)

3.3.8 Event 8: ASK

EpicBlend:	song12(10000) song11(300) song1(100) song10(100) song3(50) song5(10)

3.3.9 Event 9: REM 5 1

Playlist 1	Playlist 2	Playlist 3
song3 50 70 20 30 ~~song5 10 50 50 50~~	song4 30 10 60 70 song9 4 0 20 60 song10 100 80 0 90 song11 300 0 80 80 song12 10000 100 100 100	song1 100 99 0 50

	Heartache	Roadtrip	Blissful
	song12(100)	song12(100) song11(80) ~~song5(50)~~ song3(20)	song12(100) song10(90) song1(50) ~~song5(50)~~ song3(30)
From Playlist 1:	0	1	1
From Playlist 2:	1	2	2
From Playlist 3:	0	0	1

EpicBlend:	song12(10000) song11(300) song1(100) song10(100) song3(50) ~~song5(10)~~

3.3.10 Event 10: ASK

EpicBlend:	song12(10000) song11(300) song1(100) song10(100) song3(50)

Figure 2: Do not forget the CMPE cats for the trip—they add fun!

4 Submission

Your code will be graded automatically. Therefore, it is important that you follow the submission instructions.

First, all of your source files should be collected under **Project3/src**. Then, you should zip the **Project3** folder and rename it to <**student**$_{id}$>**.zip**. This zip file will be submitted through Moodle. Name of your main class **must** be **Project3.java**.

Your program will be compiled with the command below:

```
javac Project3/src/*.java -d ./
```

The input and output files can be in any folder. Design your code to accept the full path for file arguments. Your program will be run with the command below:

```
java Project3 <song-file> <test-case-file> <output-file>
```

Make sure your final submission compiles and runs with these commands.

5 Grading

Your grade will depend on specific tasks, which will be outlined and explained below.

	Small Case Grades	**Large Case Grades**
Test Case Group 1	6	4
Test Case Group 2	9	6
Test Case Group 3	9	6
Test Case Group 4	12	8
Test Case Group 5	12	8
Test Case Group 6	12	8

Table 6: Grading Table out of 100 points

This table shows how grades are distributed for each group. Each group has different characteristics. In addition, these features will be tested in both small and large cases.

For each test case group, there will be one small and one large test case.

In each test case group, if your code passes the small test case, you will receive points for the small category, and the same rule applies to the large category.

You can evaluate your progress in the project based on the total of 12 test cases provided. This serves as a tool for you to gain insight into your grade.

You won't be graded using the provided test cases, but new ones will be generated using the same logic and parameters.

There will be a time limit for each test case, with expected run-times provided. These durations are calculated using an Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GHz. Ensure that the run-time of your program does not exceed three times the expected run-time for each test case.

Here are the details for each group:

5.1 Test Case Group 1

In this group, all test cases include only one ASK event.

- ask_small.txt $->$ Time taken: 662ms
- ask_large.txt $->$ Time taken: 3125ms

5.2 Test Case Group 2

In this group, all test cases have only one playlist.

- one_playlist_small.txt $->$ Time taken: 789ms
- one_playlist_large.txt $->$ Time taken: 19008ms

5.3 Test Case Group 3

In all test cases within this group, the total number of songs in each playlist does not exceed 10.

- tiny_playlists_small.txt $->$ Time taken: 1635ms
- tiny_playlists_large.txt $->$ Time taken: 20080ms

5.4 Test Case Group 4

In all test cases of this group, the total number of playlists does not exceed 10.

- ten_playlists_small.txt $->$ Time taken: 836ms
- ten_playlists_large.txt $->$ Time taken: 24688ms

5.5 Test Case Group 5

In this group, all test cases involve only ADD events.

- add_small.txt $->$ Time taken: 729ms
- add_large.txt $->$ Time taken: 9242ms

5.6 Test Case Group 6

In test cases within this group, there are no additional constraints provided for evaluation. This marks the final step.

- general_small.txt $->$ Time taken: 828ms
- general_large.txt $->$ Time taken: 18669ms

5.7 Constraints for Small Test Cases

$1 \leq S, N, E \leq 100$
$1 \leq l, c_h, c_r, c_b \leq S$

5.8 Constraints for Large Test Cases

$1 \leq S, N, E \leq 10^6$
$1 \leq l, c_h, c_r, c_b \leq S$

NOTE 1: Do not focus on completing each test case group individually, as this may consume too much time. It is suggested to first consider the best solution you can think of before starting the implementation.

NOTE 2: Should your submission fail to run during our automated testing, an initial score of **0** will be assigned, necessitating the initiation of an objection process.

6 Warnings

1. This is an individual project.
2. All source codes are checked automatically for similarity with other submissions and exercises from previous years. Make sure you write and submit your own code. Any sign of cheating will be penalized by at least **-100** points at the first attempt and disciplinary action in case of recurrence.

3. You cannot include **any** external libraries to your project. **Any sorted structure** is **not allowed** including TreeSet, TreeMap, PriorityQueue. Additionally, any built-in **sorting functions** are **not allowed**. You should implement your own classes for further needs.

4. You are strongly encouraged to write and use appropriate data structures covered in lectures to achieve decent running times for large inputs. Don't forget that a significant portion of points will be given based on performance with large inputs. Therefore, using correct data structures is an integral part of this project.

5. Parallel Programming is not allowed. Directly using any parallel programming structure such as Stream and Thread will get a **0**.

6. Make sure you document your code with necessary inline comments and use meaningful variable names. Do not over-comment or make your variable names unnecessarily long. This is very important for partial grading.

7. You can add as many files as you can as long as they are in the "src" folder, and your project can be compiled as above. But the entry point of your program should be "Project3.java".

8. Make sure that the white spaces in your output are correct. You can disregard the ones at the end of the line.

9. Please use the discussion forum at Moodle for your questions and check if it has already been answered before.