

General Instructions

- This lab test carries 20 marks. The test consists of two questions numbered 1 and 2.
 - Programs should be written in *C* language.
 - Assume that all inputs are valid.
 - Sample inputs are just indicative and need not handle every input case.
 - Use of global variables is NOT permitted, unless specified otherwise.
 - The function prototypes given in the questions, if any, should not be changed.
 - The input should be read from, and the output should be printed to, the console.
 - **No clarifications regarding questions will be entertained. If there is any missing data you may make appropriate assumptions and write the assumptions clearly in the design sheet.**
 - **For both Part 1 and Part 2 of the question, students have to first write the design and then only they can proceed with the implementation.**
 - **Solve Part 2 only after submitting a solution (design and code) for Part 1.**
 - There will be a viva voce during the test.
 - Design submission:
 1. Read the question, understand the problem and write the design in pseudocode (in the shared Google document) for the indicated function(s). You are advised to **properly think about the solution before starting to write your design** to avoid wasting your time on rewrites.
 2. The design written should **clearly** convey your overall idea for the solution; the programming specific details may be changed over the course of the implementation. There will be a reduction in marks if the student writes C code instead of pseudocode.
 3. The edit permission of the shared documents for writing the design of Part 1 and Part 2 will be revoked at 2:40 PM and 5:00 PM respectively.
 4. Once designed, you should **write a program that implements your design**.
 5. If, while implementing, you realize that your design has major issues, you can ask your evaluator for permission to change your design. The evaluator will decide whether to permit the modification or not by looking at your current progress.
 6. In any case, modifications to design for Question 1 will not be permitted beyond 3:15 PM.
 - Mode of submission and timings:
 1. Part 1 Design (shared google document) - 2:40 PM
 2. Part 1 Implementation (upload the source file in EduServer) - 4:25 PM
 - The submission link will be disabled at 4:30 PM.
 3. Part 2 Design (shared google document) - 5:00 PM
 4. Part 2 Implementation (upload the source file in EduServer) - 5:25 PM
 - The submission link will be disabled at 5:30 PM.
-

- While implementing, you may use the source codes that you have previously submitted for the assignments, if you feel that it will be helpful.
- There will be some test cases that only test the correctness of specific functions and not the entire program. As such, if you are not able to complete your program, you should still make sure that your submitted code will compile and run without errors to get the marks for such test cases.
- The source code file should be named in the format

TEST<NUMBER>_<ROLLNO>_<FIRST-NAME>_<PROGRAM-NUMBER>.c

(For example, TEST2_B190001CS_LAXMAN_1.c)

The source file must be zipped and uploaded. Only zip files may be uploaded, even if they contain on a single .c file. The name of the zip file must be

TEST<NUMBER>_<ROLLNO>_<FIRST-NAME>.zip

(For example: TEST2_ B190001CS_LAXMAN.zip)

- Naming conventions **must** be strictly followed. Any deviations from the specified conventions, or associated requests after the exam, will not be considered.
- Any malpractice observed will lead to zero marks in the test. These will also be reported to the department for permission to award F grade in the course.

Mark distribution

Maximum marks – 20

- Question 1: 14 Marks (Design - 7 marks, Implementation and Test cases - 5 marks, Viva voce - 2 marks)
 - Question 2: 6 Marks (Design - 3 marks, Implementation and Test cases - 3 marks)
-

General Note:

- For the following programs, use of global variables is NOT permitted unless explicitly specified.
 - Printing of float numbers should be always rounded to two decimal points.
1. IMDb has a collection of n movies from each of the three languages English E, Hindi H and Malayalam M. You are given Movie ID M_id (string) and the Rating $rating$ (float) of movies from each language as a separate list. The details are given in the non-increasing order of their $rating$.

You have to find and print the movie/s at the first r ranking positions according to their $rating$ among all three languages by using the concept of merge.

Note: All the movies that have same $rating$ are assigned the same rank.

Your program should implement the following functions as per the function prototypes:

- $main()$ - Repeatedly read a character from $\{s, b, p, l, t\}$ and call the sub-functions appropriately until character t is encountered.
- $store_Db(Db, n)$: Read Movie ID M_id and Rating $rating$ of n movies and store it in the array Db .
- $list_Db(Db, n)$: For each movie in the array Db of length n , print the M_id and $rating$ (separated by a space), in a new line.
- $popular_rating(Db, n)$: Find and print the rating which is given to most number of movies in the array Db of length n . If there same number of movies with two different $rating$, consider the highest rating among them.
- $best_r(E, H, M, n, r)$:
 - Merge the arrays E , H and M , each of size n to a combined sorted array (if the ratings of any two movies are same, then their relative order should be preserved as in the input).
 - If no movie is present at ranking position r , return -1.
 - Otherwise, Using function $list_Db()$, print the details of the movie/s which is at ranking position from 1 to r , and return the $rating$ of the movie/s with ranking position r .

Input/Output Format

The input consists of multiple lines. Each line may contain a character from $\{s, b, a, l, t\}$ followed by zero or more integers.

- The first line contains an integer n , the number of movies in each collection.
 - Character s : Character s is followed by an integer from $\{1, 2, 3\}$
 - If character s is followed by 1, the next n lines contain M_id and $rating$ of the n movies in collection E, separated by a space.
 - If character s is followed by 2, the next n lines contain M_id and $rating$ of the n movies in collection H, separated by a space.
 - If character s is followed by 3, the next n lines contain M_id and $rating$ of the n movies in collection M, separated by a space.
-

- Character *p*: Character *p* is followed by an integer from {1, 2, 3}
 - If character *p* is followed by 1, print the rating which is given for most number of the movies in collection E.
 - If character *p* is followed by 2, print the rating which is given for most number of the movies in collection H.
 - If character *p* is followed by 3, print the rating which is given for most number of the movies in collection M.
- Character *b*: Character *b* is followed by an integer corresponding to the ranking position *r*.
 - Call function *best_r(E, H, M, n, r)*
(Note that *best_r* prints the details of the movies at ranking position 1 to *r*.)
 - Print the *rating* of the movie/s at ranking position *r*
- Character *l*: Character *l* is followed by an integer from {1, 2, 3}.
 - If character *l* is followed by 1, print the details of movies in collection E.
 - If character *l* is followed by 2, print the details of movies in collection H.
 - If character *l* is followed by 3, print the details of movies in collection M.
- Character *t*: Terminate the program.

Sample Input and Output

Input

```
5
s 1
IMDB101 9.5
IMDB003 8.7
IMDB302 5.1
IMDB112 5.1
IMDB004 2.4
s 2
IMDB110 7.8
IMDB212 7.5
IMDB654 6.5
IMDB780 6.5
IMDB389 2.4
s 3
IMDB444 9.5
IMDB761 8.7
IMDB900 8.7
IMDB297 6.7
IMDB010 5.1
b 4
l 2
p 1
t
```

Output

IMDB101 9.50
IMDB444 9.50
IMDB003 8.70
IMDB761 8.70
IMDB900 8.70
IMDB110 7.80
IMDB212 7.50
7.50
IMDB110 7.80
IMDB212 7.50
IMDB654 6.50
IMDB780 6.50
IMDB389 2.40
5.10

2. Suppose in Q.1, you are given the unsorted list of movie details from m different languages. Sort each of the lists separately in the non-increasing order of their *rating* and merge the m lists into a single sorted list. Your program must sort the lists using a modified *heap_sort* function as given below: (*More efficient designs will be awarded more marks.*)

- *heap_sort(A)* :
 - Sort the list A with the heap H implemented using an array (with starting index 0) in the following way:
 - * Each of the array elements corresponds to a node in the heap H .
 - * Root node of the heap H is at array index 0.
 - * Each node in the heap has h children, and the children of the node at index i are stored at indices $(h * i) + 1, (h * i) + 2 \dots (h * i) + h$.

Sort the m lists separately using the modified *heap_sort* function and then merge the sorted lists using the following *merge()* function:

- *merge()* : (*You can decide the function prototype based on your design.*)
 - Merge the m sorted lists into a single sorted list.
 - While doing the merge, use the heap H to find the largest element of the m lists.

Note: As m and h are constants, use them as global variables.

Input Format

- The input consists of multiple lines.
- The first line contains two integers corresponding to m and h respectively, separated by a space.
- Subsequent lines contain the details of the movies of m different languages. For each of the m languages:
 - First line contains an integer n which is the number of movies listed from that language.
 - Each of the next n lines contains a string and a float corresponding to the M_id and *rating* of the movie, separated by a single space.

Output Format

- The output contain multiple lines, corresponding to the details of the movies from m different languages in the non-decreasing order of their *rating*.
- Each line of the output should contain the M_id and *rating* of the corresponding movies, separated by a single space.

Sample Input and Output

Input

```
4 3
5
IMDB001 8.7
```

```
IMDB002 8.5
IMDB003 9.1
IMDB004 8.7
IMDB005 8.3
4
IMDB101 8.3
IMDB102 8.0
IMDB103 8.5
IMDB104 8.3
6
IMDB201 7.9
IMDB202 6.3
IMDB203 6.1
IMDB204 8.7
IMDB205 8.5
IMDB206 7.9
4
IMDB301 7.8
IMDB302 9.2
IMDB303 9.1
IMDB304 8.9
```

Output

```
IMDB302 9.20
IMDB003 9.10
IMDB303 9.10
IMDB304 8.90
IMDB001 8.70
IMDB004 8.70
IMDB204 8.70
IMDB002 8.50
IMDB103 8.50
IMDB205 8.50
IMDB005 8.30
IMDB101 8.30
IMDB104 8.30
IMDB102 8.00
IMDB201 7.90
IMDB206 7.90
IMDB301 7.80
IMDB202 6.30
IMDB203 6.10
```
