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. Air India Express is flying from Calicut to Mumbai today. n passengers have reported in each counter C_1, C_2, C_3 from 9.00 to 10.30 AM.

   You are given the Boarding pass number B_no (string) and reporting time R_time (float) of passengers of the three counters as separate lists. The details are given in the non-decreasing order of their R_time.

   Passengers are to be grouped according to their R_time. All the passengers having the same R_time are to be in the same group (a group can contain either one or more passengers). The groups are numbered $1, 2, 3, \ldots m$ as per the non-decreasing order of R_time (Group 1 consisting of passengers who arrived first).

   Using the concept of merge, print the details of passengers in group 1 to k. Your program should implement the following functions as per the function prototypes:

   - $main()$ - Repeatedly read a character from $\{r, d, p, g, t\}$ and call the sub-functions appropriately until character t is encountered.
   - $read\_details(C, n)$: Read the Boarding pass number B_no and reporting time R_time of n passengers reported at counter C and store it in the array C.
   - $display\_details(C, n)$: For each passenger in the array C of length n, print the B_no and R_time (separated by a space), in a new line.
   - $peak\_time(C, n)$: Find and print the R_time at which most number of passengers in the array C of length n have reported. If same number of passengers have reported at two different R_time, consider the earlier time.
   - $group\_g(C\_1, C\_2, C\_3, n, k)$:
     - Merge the arrays C_1, C_2 and C_3, each of size n to a combined sorted array(if the R_time of any two passengers are same, then their relative order should be preserved as in the input)
     - if no passengers exist in k^{th} group, return -1.
     - Otherwise, using the function $display\_details()$, print the details of passengers in group 1 to k, and return the R_time of the passenger/s in the group k.

Input/Output Format

The input consists of multiple lines. Each line may contain a character from $\{r, d, p, g, t\}$ followed by zero or more integers.

- The first line contains an integer n, the number of passengers in each counter.
- Character r: Character r is followed by an integer from $\{1, 2, 3\}$
  - If character r is followed by 1, the next n lines contain B_no and R_time of the n passengers of counter C_1, separated by a space.
  - If character r is followed by 2, the next n lines contain B_no and R_time of the n passengers of counter C_2, separated by a space.

  - If character r is followed by 3, the next n lines contain B_no and R_time of the n passengers of counter C_3, 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 R_time at which most of the passengers in the counter C_1 have reported.
  - If character p is followed by 2, print the R_time at which most of the passengers in the counter C_2 have reported.
  - If character p is followed by 3, print the R_time at which most of the passengers in the counter C_3 have reported.
- Character g: Character g is followed by an integer corresponding to the group number k.
  - Call function $group\_k(C\_1, C\_2, C\_3, n, k)$ (Note that $group\_k$ prints the details of passengers in group 1 to k.)
  - Print the R_time of the passenger/s in group k.
- Character d: Character d is followed by an integer from $\{1, 2, 3\}$.
  - If character d is followed by 1, print the details of passengers who reported at counter C_1.
  - If character d is followed by 2, print the details of passengers who reported at counter C_2.
  - If character d is followed by 3, print the details of passengers who reported at counter C_3.
- Character t: Terminate the program.

Sample Input and Output

Input

```
3
r 1
AIR001 9.10
AIR023 9.20
AIR090 10.10
r 2
AIR022 9.20
AIR098 10.00
AIR100 10.00
r 3
AIR078 9.45
AIR670 10.00
AIR232 10.10
g 4
d 2
p 2
t
```

Output

```
AIR001 9.10
AIR023 9.20
AIR022 9.20
AIR078 9.45
AIR098 10.00
AIR100 10.00
AIR670 10.00
10.00
AIR022 9.20
AIR098 10.00
AIR100 10.00
10.00
```

2. Suppose in Q.1, you are given the unsorted list of passengers reported at m counters. Sort each of the lists separately in the non-decreasing order of R_time 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 . . . $(h * i)$+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 passengers reported at m counters. For each of the m counters:
  - First line contains an integer n which is the number of passengers reported at that counter.
  - Each of the next n lines contains a string and a float corresponding to the B_no and R_time of a passenger, separated by a single space.

Output Format

- The output contain multiple lines, corresponding to the details of the passengers reported at m counters in the non-decreasing order of their R_time.
- Each line of the output should contain the B_no and R_time of the corresponding passengers, separated by a single space.

Sample Input and Output

Input

```
4 3
5
AIR001 9.10
```

```
AIR098 10.20
AIR067 9.55
AIR002 9.10
AIR988 10.04
4
AIR010 10.00
AIR672 9.55
AIR444 9.40
AIR445 10.22
6
AIR045 9.23
AIR047 9.55
AIR089 10.20
AIR008 10.00
AIR987 9.15
AIR777 9.40
3
AIR666 9.27
AIR766 10.21
AIR013 10.10
```

Output

```
AIR001 9.10
AIR002 9.10
AIR987 9.15
AIR045 9.23
AIR666 9.27
AIR444 9.40
AIR777 9.40
AIR067 9.55
AIR672 9.55
AIR047 9.55
AIR010 10.00
AIR008 10.00
AIR988 10.04
AIR013 10.10
AIR098 10.20
AIR089 10.20
AIR766 10.21
AIR445 10.22
```