National Institute of Technology Calicut
Department of Computer Science and Engineering
Second Semester B. Tech.(CSE)
CS1092E Program Design Laboratory
Assignment #3
Submission deadline (on or before): 19/03/2024, 08:00 PM

Naming Conventions for Submission

- Submit a single ZIP (.zip) file (do not submit in any other archived formats like .rar, .tar, .gz). The name of this file must be

  `<FIRST-NAME>_<ROLLNO>.zip`

  (Example: *LAXMAN_BxxyyyyCS.zip*). DO NOT add any other files (like temporary files, input files, etc.) except your source code, into the zip archive.

- The source codes must be named as

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

  (For example: *LAXMAN_BxxyyyyCS_1.c*). If you do not conform to the above naming conventions, your submission might not be recognized by our automated tools, and hence will lead to a score of 0 marks for the submission. So, make sure that you follow the naming conventions.

Standard of Conduct

- Violation of academic integrity will be severely penalized. Each student is expected to adhere to high standards of ethical conduct, especially those related to cheating and plagiarism. Any submitted work **MUST BE an individual effort**. Any academic dishonesty will result in zero marks in the corresponding exam or evaluation and will be reported to the department council for record keeping and for permission to assign F grade in the course. The department policy on academic integrity can be found at: http://cse.nitc.ac.in/sites/default/files/Academic-Integrity_new.pdf.

General Instructions

- Programs should be written in C language.
- Check your programs with sufficiently large values of inputs with in the range as specified in the question.
- Global and/or static variables should not be used in your program.

Q1 Write a C program that takes N names of cars and its corresponding mileage as input and sorts them in descending order of mileage using heapsort. Use structures to store the car data.

Input Format
The first line of the program contains the integer N which is the number of elements to be stored. ($1 \leq N \leq 1000$).
The next N lines contain the name of the car and its corresponding mileage separated by a colon character. Note that the car names could also have spaces in between. You can assume that the car names will never have a colon character.

Output Format:
N lines where the car data is sorted in descending order of mileage.

Sample Input :
3
Maruti Alto k10:33
Maruti Wagon R:34
Maruti Celerio:35

Sample Output:
Maruti Celerio:35
Maruti Wagon R:34
Maruti Alto k10:33

Q2 - Design a job dispatcher program using a priority queue implemented with a binary heap. The program selects a job to execute based on its priority from a pool of jobs. Each job has a unique job-id: ID and a priority: p, both of which are integers. Only one job can execute at a time, and the pool of jobs can be updated with more jobs. New jobs are added to the pool upon arrival and removed when scheduled for execution. The job dispatcher selects the job with the highest priority for execution and removes it from the pool. (Note: You may assume that the job pool will contain at most 20 jobs at a time.)

Implement the following functions:

main(): Repeatedly reads an input character from the menu list through the terminal and executes menu-driven operations accordingly. The menu list is ['a', 's', 'n', 'r', 'd', 'e']. The program ends when 'e' is input.
add(Que, ID, p): Adds a new job with job-id ID and priority p to the pool of jobs..
schedule(Que): Schedules the job with the highest priority from the pool of jobs for execution, removes it from the pool, and prints its details. Print "-1" if the pool is empty.

next_job(Que): Displays the ID of the next job to be executed, which will be the one with the highest priority from the pool of jobs. The job is not removed from the pool. Print "-1" if the queue is empty.
replace_priority(Que, ID, np): Replaces the priority of job ID in the pool of jobs with the new priority np. Print "-1" if the ID is not in the pool of jobs.
display(Que): Prints the details (ID and p) of each job in the pool of jobs in the level-wise order of the jobs stored in the heap. Print "-1" if the pool is empty.

(Note: Ensure that heap property will be maintained while adding a new job, scheduling a job, and replacing priority.)

Input format:

Each line of input contains a character from the menu list ['a', 's', 'n', 'r', 'd', 'e'], followed by at most two integers depending on the menu function (either nothing, or two).
Input 'a' followed by two positive integers, representing job-id as ID and priority as p, respectively, performs the function add(Que, ID, p).
Input ‘s’ performs the function schedule(Que).
Input ‘n’ performs the function next_job(Que).
Input 'r' followed by two positive integers, representing job-id as ID and a new priority as np, respectively, performs the function replace_priority(Que, ID, np).
Input ‘d’ performs the function display(Que).

Output format:

Each line contains an integer or two integers representing ID and priority, respectively.
Lines may also contain the integer "-1" depending on the input.

Sample input:
a 1 40
a 2 30
a 3 50
a 4 20
s
n
d
r 2 35
d
e

Sample Output:
3
1
1 40

2 30
4 20
1 40
2 35
4 20

Q3 - An arithmetic expression is a cascade of expressions of the format <number><operator><number> (where the operators are +,-,*,/,^) with parentheses to emphasize the importance of certain arithmetic operations. An arithmetic expression has balanced parentheses pairs when for every open parentheses '(', there has to be an associated closing parentheses ')' and the total number of open and close parentheses has to be equal. For example, the expression (4+3)-2+((5-3)*2+4)+8-(3+1) is a balanced expression as every open parenthesis has an associated closing parenthesis.
Write a program that takes N arithmetic expressions as input and determines whether each expression has balanced parentheses pairs or not.

Input Format:
The first line consists of the number of test cases 'N'.
The next N lines would input the 'N' arithmetic expressions.

Output Format:
A single line providing the number of input expressions that have balanced paranthesis.

Sample Input:
4
((4+3)-2+((5-3)*2+4)+8-(3+1)
(4+3)-2+((5-3)*2+4)+8-(3+1)
(4+3)-2+((5-3)*2+4)+8-(3+1))(
(4+3)-(2+((5-3)*2+4)+8)-(3+1)

Sample Output :
2

Explanation:
The expression ((4+3)-2+((5-3)*2+4)+8-(3+1) is not balanced since there exists an open parenthesis which doesn't have a matching closing parenthesis. The expression (4+3)-2+((5-3)*2+4)+8-(3+1) is balanced. The expression (4+3))-2+((5-3)*2+4)+8-(3+1))(Is not balanced since there exists an open parenthesis at the end with no matching closing parenthesis. The expression (4+3)-(2+((5-3)*2+4)+8)-(3+1) is balanced. Out of 4 input expressions, 2 expressions were balanced, making the corresponding output as 2.