Seneca College

Feb 07, 2020

Applied Arts & Technology

SCHOOL OF COMPUTER STUDIES

JAC444 Demo & Final Code Due date Oct 14, 2020

Workshop 4

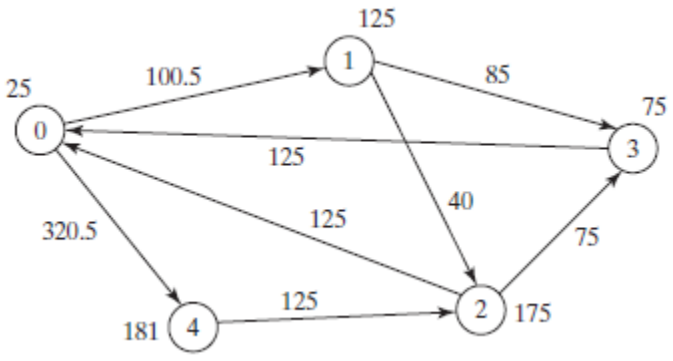
Notes:

i. Each task should be presented during the lab, demo worth 70% of the workshop marks and code uploading worth the other 30%.
ii. Make sure you have all security and check measures in place, like wrong data types etc., implement proper Exception Handling techniques
iii. Given output structure is just for student to have a glimpse what the output can look, students are free to design the output better in any way.
iv. The final code as zip should be submitted by the midnight to avoid late penalties which are 10% each day late.
v. The workshop will be marked as follows:
    a. Proper naming of the class/s expected.
    b. Proper documentation for all the class/s, method/s etc. used.
    c. Clear naming for the variables, class/s, methods expected.
    d. Output should be clear, and sentences should make sense.
    e. Clear all debugging fields, data, line etc. used in the code.

Other inputs can be given during demo, so make sure you test your program properly.

Task 1:

Banks lend money to each other. In tough economic times, if a bank goes bankrupt, it may not be able to pay back the loan. A bank's total assets are its current balance plus its loans to other banks. The diagram below shows five banks. The banks' current balances are 25, 125, 175, 75, and 181 million dollars, respectively. The directed edge from node 1 to node 2 indicates that bank 1 lends 40 million dollars to bank 2.



Banks lend money to each other

If a bank's total assets are under a certain limit, the bank is unsafe. The money it borrowed cannot be returned to the lender, and the lender cannot count the loan in its total assets. Consequently, the lender may also be unsafe, if its total assets are under the limit.

Write a program to find all the unsafe banks. Your program reads the input as follows.

1. It first reads two integers **n** and **limit**, where **n** indicates the number of banks and **limit** is the minimum total assets for keeping a bank safe from the user.
2. It then reads **n** lines that describe the information for **n** banks with IDs from **0** to **n-1**.

The first number in the line is the bank's balance, the second number indicates the number of banks that borrowed money from the bank, and the rest are pairs of two numbers. Each pair describes a borrower. The first number in the pair is the borrower's ID and the second is the amount borrowed.
For example, the input for the five banks in above picture is as follows (**note that the limit is 201**):

Number of banks: 5
Minimum asset limit: 201
Bank # 0 → Balance: 25 → Number of banks Loaned: 2 → Bank ID: 1 → Amount: 100.5 → Bank ID: 4 → Amount: 320.5
Bank # 1 → Balance: 125 → Number of banks Loaned: 2 → Bank ID: 2 → Amount: 40 → Bank ID: 3 → Amount: 85
Bank # 2 → Balance: 175 → Number of banks Loaned: 2 → Bank ID: 0 → Amount: 125 → Bank ID: 3 → Amount: 75
Bank # 3 → Balance: 75 → Number of banks Loaned: 1 → Bank ID: 0 → Amount: 125
Bank # 4 → Balance: 181 → Number of banks Loaned: 1 → Bank ID: 2 → Amount: 125

The total assets of bank 3 are (75 + 125), which is under 201, so bank 3 is unsafe. After bank 3 becomes unsafe, the total assets of bank 1 fall below (125 + 40). Thus, bank 1 is also unsafe.
Note: Program should take inputs from the user like Number of banks, Minimum asset limit and then all other inputs

The output of the program should be
Unsafe banks are 3 and Bank 1

Task 2:

Write a hangman game that randomly generates a word and prompts the user to guess one letter at a time, as shown in the sample run.

- Each letter in the word is displayed as an asterisk. When the user makes a correct guess, the actual letter is then displayed.
- When the user finishes a word, display the number of misses and ask the user whether to continue to play with another word.
- If the user missed the same letter more than one time give extra hint to the player like "You've already tried this letter try another letter". Don't count twice the same letter which is missed by the user.
- Have minimum 10 words of your choice in some array, don't create new words using alphabets.