Nhớ những cái father thì phải tìm parent của nó

# DE1: PE [#CSD201](https://www.facebook.com/hashtag/csd201?__eep__=6&__cft__%5b0%5d=AZW9X9NaZc61Tqnhn40vCm8372PJq2Zo-M52Th14aVwVOj1sAl2Tz9F9H0b8TZsKGF63J26-JIu2jIKaD7tLVInE8dBqrUrIzbXh7RNs0aAy0faWg28aLrfqHPvwIapV2KIc2BwXJSLj4T_qVK84Vzwv&__tn__=*NK-R) SU23 Hola Ca 7h30 Paper No 4

## Question 1: (4 marks)

In this question you should complete some methods in MyList.java file. The class Boat with 3 data members: sea, sail and paddle is given and you do not need to edit it. The MyList class is a linked list of Boat objects. The following methods should be completed:

- void addLast(String xSea, int xSail, int xPaddle) - check if xSea.charAt(0) == 'A' then do nothing, otherwise add new node with sea=xSea, sail=xSail, paddle=xPaddle to the end of the list. (sail and paddle can get arbitrary, even negative values).

- void f1() - Do not edit this method. Your task is to complete the addLast(...) method above only. Output in the file f1.txt must be the following: (8,5,3) (C,6,5) (D,2,4) (E,7,9) (F,4,-7) (G,-3,2)

- void f2()-There are 3 given Boat objects x, y, z in this function. Suppose the list contains at least 3 elements. Write statements to insert x, y and z to the list so that x, y, z will be at **positions 1, 3 and 4 (The head's position is 0).** Output in the file f2.txt must be the following:

(C,9,8) (D,6,3) (E,8,5) (F,5,4) (I,4,9) (J,3,7)

(C,9,8) (X,1,2) (D,6,3) (Y,2,3) (Z,3,4) (E,8,5) (F,5,4) (I,4,9) (I,3,7)

- void f3() - Move the second node having **maximum paddle** to the tail (thus if there is only one maximum paddle then do nothing). Output in the file f3.txt must be the following:

(C,8,6) (D,3,9) (E,9,2) (F,5,9) (G,9,7) (H,6,8) (I,7,3)

(C,8,6) (D,3,9) (E,9,2) (G,9,7) (H,6,8) (I,7,3) (F,5,9)

- void f4() - Suppose the list contains at least 7 elements. Sort the 5 elements from position 2 to position 6 ascendingly by paddle (The head's position is 0). The content of the output file f4.txt must be the following:

(C,9,8) (D,11,12) (E,10,11) (F,1,19) (I,7,9) (J,6,8) (K,5,6) (L,4,5) (M,3,4)

(C,9,8) (D,11,12) (K,5,6) (J,6,8) (I,7,9) (E,10,11) (F,1,19) (L,4,5) (M,3,4)

## Question 2: (4 marks)

In this question you should complete some methods in BSTree.java file.

The class Boat with 3 data members: sea, sail and paddle is given and you do not need to edit it. The BSTree class is a binary search tree of Boat objects. The variable **sail** is the key of the tree, thus it must be unique. The following methods should be completed:

void insert(String xSea, int xSail, int xPaddle) - check if xSea.charAt(0)== 'B' then do nothing, otherwise insert new Boat object with sea=xSea, sail=xSail, paddle-xPaddle to the tree (sail and paddle can get arbitrary, even negative values).

void f1()- Do not edit this method. Your task is to complete the **insert**(...) method above only. Output in the file f1.txt must be the following:

(A,7,9) (C,4,3) (D,8,6) (E,2,5) (Y,6,-7) (F,-6,7)

(F,-6,7) (E,2,5) (C,4,3) (Y,6,-7) (A,7,9) (D,8,6)

- void f2() - Perform breadth-first traversal from the root but display to file f2.txt nodes with paddle>5 only. Hint: Copy the function breadth(...) to function breadth2(...) and modify it. Output in the file f2.txt must be the following:

(C,8,2) (D,6,1) (E,9,4) (F,2,-1) (G,7,3) (H,10,8) (I,1,7) (J,3,9) (K,-1,5) (L,5,10) (M,4,6)

(H,10,8) (I,1,7) (J,3,9) (L,5,10) (M,4,6)

- void f3() - Perfom in-order traversal from the root and find the second node p having left child. If such node does not exist then do nothing, otherwise delete the node q = p.left by copying. Output in the file f3.txt must be the following:

(C,8,2) (D,6,1) (E,9,4) (F,2,-1) (G,7,3) (H,10,8) (I,1,7) (J,3,9) (K,-1,5) (L,5,10) (M,4,6)

(C,8,2) (D,6,1) (E,9,4) (F,2,-1) (G,7,3) (H,10,8) (K,-1,5) (J,3,9) (L,5,10) (M,4,6)

- void f4() - Perfom **in-order** traversal from the root and find the **second node p having left child**. If such node does not exist then do nothing, otherwise **rotate p to right**. Output in the file f4.txt must be the following:

(C,8,2) (D,6,1) (E,9,4) (F,2,-1) (G,7,3) (H,10,8) (I,1,7) (J,3,9) (K,-1,5) (L,5,10) (M,4,6)

(C,8,2) (D,6,1) (E,9,4) (I,1,7) (G,7,3) (H,10,8) (K,-1,5) (F,2,-1) (J,3,9) (L,5,10) (M,4,6)

## Question 3: (2 marks)

In this question you should complete some methods in Graph.java file.

The class Graph is the implementation of a graph. The following methods should be completed:

- void f1()-Perform breadth-first traversal (to the file f1.txt) from the vertex i=4 (the vertex E) but display 5 vertices from the 2nd vertex to the 6th vertex only. Hint: copy breadth(...) to breadth2(...) and modify the latter one. Content of the output file f1.txt must be:

EBHAICDGF

BHAIC

void f2()-Apply the Dijkstra's shortest path algorithm to find

(1) the shortest path from vertex 0 (A) to vertex 6 (G),

then (2) from vertext 2 (C) to vertex 5 (F).

Write 3 lines to the file f2.txt:

line 1 contains the last 3 vertices selected into the set S for shortest path (1),

line 2 contains labels of vertices in the line 1 correspondingly,

line 3 contains vertices in shortest path (2).

(Note that in the weighted matrix, the value 99 is considered as infinity). Output in the file f2.txt must be the following:

D F G

19 24 29

C E D F

# DE 2 – FA22 == done (cop bai chi Dung)

## Question 1: (4 marks)

In this question you should complete some methods in MyList.java file. The class Bike with 3 data members: brand, color and weight is given and you do not need to edit it. The MyList class is a linked list of Bike objects. The following methods should be completed:

void **addLast**(String xBrand, int xColor, int xWeight) - check if xBrand.charAt(0) == 'A' then do nothing, otherwise add new node with brand=xBrand, color=xColor, weight=xWeight to the end of the list. (color and weight can get arbitrary, even negative values).

* void f1()-Do not edit this method. Your task is to complete the addLast(...) method above only. Output in the file f1.txt must be the followeight:

(B,5,3) (C,6,5) (D,2,4) (E,7,9) (F,4,-7) (G,-3,2)

* void f2()-There are 2 given Bike objects x, y in this function. Suppose the list contains at least 5 elements. Write statements to insert x and y to the list so that **y will be the 3-rd**, x will be before the last node. Output in the file f2.txt must be the following:

(C,9,8) (D,6,3) (E,8,5) (F,5,4) (I,4,9)

(C,9,8) (D,6,3) (Y,3,4) (E,8,5) (F,5,4) (X,1,2) (I,4,9)

* void f3() Remove the second node having weight<6. Output in the file f3.txt must be the following:

(C,8,6) (D,3,5) (E,9,2) (F,5,8) (G,9,7) (H,6,8) (I,7,3)

(C,8,6) (D,3,5) (F,5,8) (G,9,7) (H,6,8) (I,7,3)

* void f4()-Sort from (first) min weight to the end ascendingly by weight. The content of the output file f4.txt must be the following:

(C9,8) (D,3,7) (E,10,2) (F,1,9) (I,6,4) (J,11,5) (K,7,6)

(C,9,8) (D,3,7) (E,10,2) (I,6,4) (J,11,5) (K,7,6) (F,1,9)

## Question 2: (4 marks)

In this question you should complete some methods in BSTree.java file.

The class Bike with 3 data members: brand, color and weight is given and you do not need to edit it. The BSTree class is a binary search tree of Bike objects. The variable **weight** is the key of the tree, thus it must be unique. The following methods should be completed:

void insert(String xBrand, int xColor, int xWeight) - check if **xBrand.charAt(0) == 'A'** then do nothing, otherwise insert new Bike object with brand=xBrand, color=xColor, weight=xWeight to the tree (color and weight can get arbitrary, even negative values).

* void f1() - Do not edit this method. Your task is to complete the insert(...) method above only. Output in the file f1.txt must be the following:

(B,9,4) (C,4,3) (D,8,6) (Y,6,-7) (E,2,5) (F,-6,7)

(Y,6,-7) (C,4,3) (B,9,4) (E,2,5) (D,8,6) (F,-6,7)

* void f2()-Perform pre-order traversal from the root but display to file f2.txt nodes with color<6 only. Hint: Copy the function preOrder(...) to function preOrder2(...) and modify it. Output in the file f2.txt must be the following:

(C,8,2) (D,6,1) (E,9,4) (F,2,3) (G,7,8) (H,1,7) (J,5,5) (K,4,6) (I,3,9)

(F,2,3) (H,1,7) (J,5,5) (K,4,6) (I,3,9)

* void f3() - Suppose p is the 5th node when performing the post-order traversal of the tree. Calculate number of nodes in the sub-tree with root p and suppose this number is k, then set p.info.color = k. Output in the file f3.txt must be the following:

(D,6,1) (F,2,3) (K,4,6) (J,5,5) (H,1,7) (J,3,9) (G,7,8) (E,9,4) (C,8,2)

(D,6,1) (F,2,3) (K,4,6) (J,5,5) (H,3,7) (J,3,9) (G,7,8) (E,9,4) (C,8,2)

* void f4()-Perform the post-order traversal of the tree and find the 2nd node having left son. If the node found is p, then rotate p to right. Output in the file f4.txt must be the following:

(D,6,1) (F,2,3) (K,4,6) (J,5,5) (H,1,7) (J,3,9) (G,7,8) (E,9,4) (C,8,2)

(D,6,1) (F,2,3) (K,4,6) (J,5,5) (J,3,9) (G,7,8) (H,1,7) (E,9,4) (C,8,2)

## Question 3: (2 marks)

In this question you should complete some methods in Graph.java file.

The class Graph is the implementation of a graph. The following methods should be completed:

* void f1()-Perform depth-first traversal (to the file f1.txt) from the vertex i-2 (the vertex C) but display 6 vertices from the 2nd vertex to the 7th vertex only. Hint: copy depth(...) to depth2(...) and modify the latter one. Content of the output file f1.txt must be:

CABEHIDGF

ABEHID

* void f2() - Apply the Dijkstra's shortest path algorithm to find   
  (1) the shortest path from **vertext 2 (C) to vertex 5 (F),** then (2) from vertex 0 (A) to vertex 6 (G).   
  Write 2 lines to the file f2.txt:  
  line 1 contains the first and the **last vertices and shortest distance in** (1),   
  line 2 contains the last 4 vertices selected into the set S with their labels in (2). (Note that in the weighted matrix, the value 99 is considered as infinity). Output in the file f2.txt must be the following:

C->F:12

E-15 D-19 F-24 G-29

# DE3 – SP23 ==DONE

## Question 1: (4 marks)

In this question you should complete some methods in MyList.java file. The class Bike with 3 data members: brand, color and weight is given and you do not need to edit it. The

MyList class is a linked list of Bike objects. The following methods should be completed:

void **addLast**(String xBrand, int xColor, int xWeight) - check if xBrand.charAt(0) == 'B' then do nothing, otherwise add new node with brand=xBrand, color=xColor, weight-xWeight to the end of the list. (color and weight can get arbitrary, even negative values).

* void f1()-Do not edit this method. Your task is to complete the addLast(...) method above only. Output in the file f1.txt must be the following:

(A,9,8) (C,6,5) (D,2,4) (E,7,9) (F,4,-7) (G,-3,2)

* void f2()-There are 2 given Bike objects x, y in this function. Suppose the list contains at least 3 elements. Write statements to insert x and y to the list so that x will be the 2nd, y will be the 5th node. Output in the file f2.txt must be the following:

(C,9,8) (D,6,3) (E,8,5) (F,5,4) (I,4,9) (I,3,7)

(C,9,8) (X,1,2) (D,6,3) (E,8,5) (Y,3,4) (F,5,4) (I,4,9) (I,3,7)

* void f3() - Remove the second node having min color (thus if there is only one min color then do nothing). Output in the file f3.txt must be the following:

(C,8,6) (D,3,8) (E,9,2) (F,5,8) (6,3,7) (H,6,8) (I,7,3)

(C,8,6) (D,3,8) (E,9,2) (F,5,8) (H,6,8) (I,7,3)

* void f4()-Suppose the list contains at least 7 elements. Sort the **first 4 elements ascendingly** and the **last 3 elements descendingly** by color. The content of the output file f4.txt must be the followeight:

(C,9,8) (D,3,2) (E,10,7) (F,1,9) (I,6,4) (J,11,5) (K,7,6) (F.1,9) (D,3,2)

(C,9,8) (E,10,7) (J,11,5) (K,7,6) (I,6,4)

## Question 2: (4 marks)

In this question you should complete some methods in BSTree.java file.

The class Bike with 3 data members**: brand, color and weight** is given and you do not need to edit it. The BSTree class is a binary search tree of Bike objects. The variable **weight** is the key of the tree, thus it must be unique. The following methods should be completed:

void **insert**(String xBrand, int xColor, int xWeight) - check if **xBrand.charAt(0) == 'A'** then do nothing, otherwise insert new Bike object with brand=xBrand, color=xColor, weight=xWeight to the tree (color and weight can get arbitrary, even negative values).

* void f1() - Do not edit this method. Your task is to complete the insert(...) method above only. Output in the file f1.txt must be the following:

(8,9,4) (C,4,3) (D,8,6) (Y,6,-7) (E,2,5) (F,-6,7)

(Y,6,-7) (C,4,3) (8,9,4) (E,2,5) (0,8,6) (F,-6,7)

* void f2()-Perform **post-order traversal** from the root but display to file f2.txt nodes with **color<7 only**. Hint: Copy the function **postOrder**(...) to function postOrder2(...) and modify it. Output in the file f2.txt must be the following:

(F,2,-1) (D,6,1) (6,7,3) (M,4,6) (K,-1,5) (I,1,7) (L,5,10) (U,3,9) (H,10,8) (E,9,4) (C,8,2)

(F,2,-1) (D,6,1) (M,4,6) (K,-1,5) (I,1,7) (L,5,10) (I,3,9)

* void f3() - Perfom **breadth-first traversal** from the root and find the second node p **having 2 children**. If such node does not exist then do nothing, otherwise **delete the node q = p.right by copying**. Output in the file f3.txt must be the following:

(C,8,2) (D,6,1) (E,9,4) (F,2,-1) (G,7,3) (H,10,8) (I,1,7) (J,3,9) (K,-1,5) (I,5,10) (M,4,6)

(C,8,2) (0,6,1) (E,9,4) (F,2,-1) (G,7,3) (I,1,7) (K,-1,5) (J,3,9) (M,4,6) (L,5,10)

* void f4() - Find the node p having **maximum weight**. Suppose f is the father of p. Check if f is not null then rotate f about p. Output in the file f4.txt must be the following

(C,8,2) (D,6,1) (E,9.4) (F,2,-1) (G,7,3) (H,10,8) (I,1,7) (J,3,9) (K,-1,5) (L,5,10) (M,4,6)

(C,8,2) (D,6,1) (E,9,4) (F,2,-1) (6,7,3) (H,10,8) (I,1,7) (L,5,10) (K,-1,5) (J,3,9) (M,4,6)

## Question 3: (2 marks)

In this question you should complete some methods in Graph.java file.

The class Graph is the implementation of a graph. The following methods should be completed:

* void f1()-Perform depth-first traversal (to the file f1.txt) from the vertex i=3 (the vertex D) but display 5 vertices from the 3rd vertex to the 7th vertex only. Hint: copy depth(...) to depth2(...) and modify the latter one. Content of the output file f1.txt must be:

DABEHICGF

BEHIC

* void f2()-Apply the Dijkstra's shortest path algorithm to find  
  (1) the shortest path from vertext 0 (A) to vertex 6 (G)  
  then (2) from vertex 1 (8) to vertex 5 (F).   
  Write 3 lines to the file f2.txt:   
  line 1 contains the last 3 vertices selected into the set S with their labels in (1),   
  line 2 contains the 1st, 3rd and last vertices in shortest path (1),   
  line 3 contains vertices in shortest path (2).   
  (Note that in the weighted matrix, the value 99 is considered as infinity). Output in the file f2.txt must be the following:

D 19 F 24 G 29

ACG

BCEDF

# DE4 – SU22 ==DONE

## Question 1: (4 marks)

In this question you should complete some methods in MyList.java file.

The class Bird with 3 data members**: type, rate and wing** is given and you do not need to edit it. The MyList class is a linked list of Bird objects. The following methods should be completed:

void **addLast**(String Type, int xRate, int xWing) - check if **xType.charAt(0) == 'B'** then do nothing. otherwise add new node with type-xType, rate-xRate, wing-Wing to the end of the list. (rate and wing can get arbitrary, even negative values).

* void f1()-Do not edit this method. Your task is to complete the addLast(...) method above only. Output in the file f1.txt must be the following

(A,9,8) (C,6,5) (D,2,4) (E,7,9) (F,4,-7) (G,-3,2)

* void f2() -There are 2 given Bird objects x, y in this function. Suppose the list contains at least 5 elements. Write statements to insert x and y to the list so that **x will be the 4th, y will be the 6th node.** Output in the file f2.txt must be the following:

(C,9,8) (D,6,3) (E,8,5) (F.5,4) (I,4,9)

(C,9,8) (D,6,3) (E,8,5) (X,1,2) (F,5,4) (Y,3,4) (I,4,9)

* void f3()-Find the second node having rate<6 then change its **wing** to 99. Output in the file f3.txt

must be the following:

(C8,6) (D,3,5) (E,9,2) (F,5,8) (G,9,7) (H,6,8) (1.7,3)

(C8,6) (D,3,5) (E,9,2) (F,5,99) (G,9,7) (H,6,8) (I,7,3)

* void f4()-Sort from beginning to the first max rate ascendingly rate. The content of the output file f4.txt must be the following:

(C,1,2) (D,10,3) (E,2,15) (F,11,6) (I,6,14) (J,11,15) (K,7,9)

(C,1,2) (E,2,15) (D.10,3) (F.11.6) (I,6,14) (3,11,15) (K,7,9)

## Question 2: (4 marks)

In this question you should complete some methods in BSTree.java file.

The class Bird with 3 data members: **type, rate and wing** is given and you do not need to edit it. The BSTree class is a binary search tree of Bird objects. The variable **rate is the key** of the tree, thus it must be unique. The following methods should be completed:

void insert(String Type, int xRate, int xWing) - check if **xType.charAt(0) == 'B'** then do nothing. otherwise insert new Bird object with type=xType, rate=xRate, wing-xWing to the tree (rate and ng can get arbitrary, even negative values). Wing

* void f1()- Do not edit this method. Your task is to complete the insert(...) method above only.

Output in the file f1.bxt must be the following:

(A,7,9) (C.4,3) (D,8,6) (E,2,5) (Y.6,-7) (F,-6,7)

(F-6,7) (E,2,5) (C.4,3) (Y,6,-7) (A,7,9) (D,8,6)

* void f2() - Perform breadth-first traversal from the root but display to file f2.txt nodes with wing>4 only. Hint: Copy the function breadth(...) to function breadth2(...) and modify it. Output in the file f2.txt must be the following:

(C,8,2) (D,6,1) (E,9,4) (F,2,3) (G,7,8) (H.1.7) (I,3,9) (J,5,5) (K,4,6)

(G,7,8) (H,1,7) (I,3,9) (J,5,5) (K,4,6)

* void f(3)-Suppose p is the 4-th node when performing the post-order traversal of the tree and **f the father of p**. **Delete the node f by copying**. Output in the file f3.txt must be the following

(H,2,7) (K,4,6) (J,5,5) (I,3,9) (F,2,3) (G,7,8) (D,6,1) (E,9.4) (C,8,2)

(K,4,6) (J,5,5) (I,3,9) (H,1,7) (G,7,8) (D,6,1) (E,9,4) (C,8,2)

* void f4()-Suppose p is the 4-th node when performing the post-order traversal of the tree, Calculate the height of sub-tree with root p and suppose this height is k, then **set p.info wing k**. Output in the file f4.txt must be the following:

(H,1,7) (K,4,6) (1.5,5) (I,3,9) (F,2,3) (G,7,8) (D,6,1) (E,9.4) (C,8,2)

(H,1,7) (K,4,6) (J,5,5) (I,3,3) (F,2,3) (G,7,8) (D,6,1) (E,9,4) (C,8,2)

## Question 3: (2 marks)

In this question you should complete some methods in Graph.java file.

The class Graph is the implementation of a graph. The following methods should be completed:

* void f1()-Perform breadth first traversal (to the file f1.txt) from the vertex i=2 (the vertex C) but display 5 vertices from the 2nd vertex to the 6th vertex only. Hint: copy breadth...) to breadth2...) and modify the latter one. Content of the output file f1.txt must be:

C A H B D E I G F

A H B D E

* void f2()-Apply the Dijkstra's shortest path algorithm to find   
  (1) the shortest path from vertext 2 (C) to vertex 5 (F),   
  then (2) from vertex 1 (B) to vertex 6 (G).   
    
  Write 3 lines to the file 12.txt:   
  line 1 contains **vertices** in shortest path (1),   
  line 2 contains the **shortest distance** in (1),   
  line 3 contains **first 3 vertices** selected into the set 5 in (2).  
  (Note that in the weighted matrix, the value 99 is considered as infinity). Output in the file f2.txt must be the following:

C E D F

12

B C H

# DE6

## Quesion 1: (4 marks)

(Do not pay attention to real meaning of objects, variables and their values in the questions below). In this question you should complete some methods in MyList.java file.

The class Ball with 3 data members: maker, type and radius is given and you do not need to edit it. The MyList class is a linked list of Ball objects. The following methods should be completed:

void addLast(String xMaker, int xType, int xRadius) - check if xMaker.charAt(0) == 'B' then do nothing, otherwise add new node with maker-xMaker, type=xType, radius=xRadius to the end of the list. (type and radius can get arbitrary, even negative values).

* void f1(0) Do not edit this method. Your task is to complete the addLast(...) method above only. Output in the file fl.txt must be the following:

(A,9,8) (C,6,5) (D,2,4) (E,7,9) (F,4,7)

* void f2() -There are 2 given Ball objects x, y in this function. Suppose the list contains at least 5 elements. Write statements to insert x and y to the list so that x will be the 3rd, y will be the 4th element in the list. Output in the file f2.txt must be the following:

(C,9,8) (D,6,3) (E,8,5) (F,5,4) (I,4,9)

(C,9,8) (D,6,3) (X,1,2) (Y,3,4) (E,8,5) (F,5,4) (I,4,9)

* void f30) - Suppose the list contains at least 4 elements. **Remove the second node having maximum type** (thus if only one node having maximum type then do nothing). Output in the file f3.txt must be the following:

(C,8,6) (D,3,5) (E,9,2) (F,5,6) (G,9,7) (H,6,2) (I,7,8)

(C,8,6) (D,3,5) (E,9,2) (F,5,6) (H,6,2) (I,7,8)

* void f4() Suppose the list contains at least 4 elements. **Sort the first 4 elements descendingly by type.** The content of the output file f4.txt must be the following:

(C,2,1) (D,12,2) (E,3,11) (F,4,3) (I,5,4) (J,6,5) (K,7,6)

(D,12,2) (F,4,3) (E,3,11) (C,2,1) (I,5,4) (J,6,5) (K,7,6)

## Quesion 2: (4 marks)

In this question you should complete some methods in BSTree.java file. The class Ball with 3 data members: maker, type and radius is given and you do not need to edit it. The BSTree class is a binary search tree of Ball objects. The variable **type** is the key of the tree, thus it must be unique. The following methods should be completed:

void insert(String xMaker, int xType, int xRadius) - check if xMaker charAt(0) == 'B' then do nothing, otherwise insert new Ball with maker-xMaker, type=xType, **radius**=xRadius to the tree (type and radius can get arbitrary, even negative values).

* void f1() - Do not edit this method. Your task is to complete the insert(...) method above only. Output in the file fl.txt must be the following:

(A,7,9) (C,4,3) (D,8,6) (E,2,5) (F,6,7)

(E,2,5) (C,4,3) (F,6,7) (A,7,9) (D,8,6)

* void f2() - Perform post-order traversal from the root but display to file f2.txt nodes with **radius< 5 only**. Hint: Copy the function postOrder(...) to function postOrder2(...) and modify it. Output in the file f2.txt must be the following:

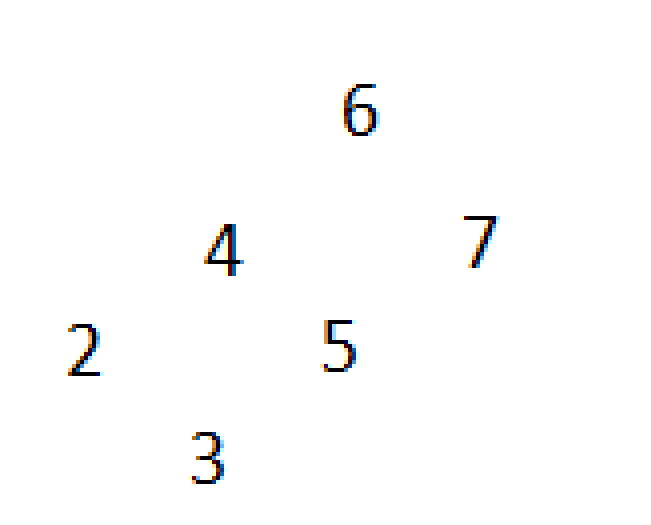
(E,2,8) (I,5,4) (H,6,3) (F,4,5) (G,8,7) (D,7,2) (C,3,6)

(I,5,4) (H,6,3) (D,7,2)

* void f3() - Perform in-order traversal and find the first node p having both 2 sons, then **delete the node p by copying**. Output in the file f3.txt must be the following:

(C,6,3) (D,4,2) (E,7,5) (F,2,4) (G,5,7) (H,3,6)

(C,6,3) (H,3,6) (E,7,5) (F,2,4) (G,5,7)



* void f4() - Perform in-order traversal and find the node p having both 2 sons, then rotate p to left. Output in the file f4.txt must be the following:

(C,6,3) (D,4,2) (E,7,5) (F,2,4) (G,5,7) (H,3,6)

(C,6,3) (G,5,7) (E,7,5) (D,4,2) (F,2,4) (H,3,6)

## Quesion 3: (2 marks)

In this question you should complete some methods in Graph.java file. The class Graph is the implementation of a graph. The following methods should be completed: void f1() - Perform depth-first traversal (to the file fl.txt) from the vertex i=3 (the vertex D) but display 5 vertices from the 2nd vertex to the 6th vertex only. Hint: copy depth(...) to depth2(...) and modify the latter one. Content of the output file fl.txt must be:

DABEHICGF

ABEHI

void f2()- Apply the Dijkstra's shortest path algorithm to find (1) the shortest path from vertext 2 (C) to vertex 5 (F), then (2) from vertex 0 (A) to vertex 6 (G). Write 3 lines to the file f2.txt: line I contains vertices in shortest path (1), line 2 contains the last 3 vertices selected into the set S with their labels in (2), line 3 contains vertices in shortest path (2). (Note that in the weighted matrix, the value 99 is considered as infinity). Output in the file f2.txt must be the following:

CED F

D,19 F,24 G,29

ABCED G

# DE8

## Question 1: (4 marks)

Do not pay attention to real meaning of objects, variables and their values in the questions below.

In this question you should complete some methods in MyList.java file. The class Castor with 3 data members: place, depth and type is given and you do not need to edit it. The MyList class is a linked list of Castor objects. The following methods should be completed:

void addLast(String xPlace, int xDepth, int xType)-check if xPlace.charAt(0)=='A'then do nothing otherwise add new node with place=xPlace, depth=xDepth, type=xType to the end of the list. (depth and type can get arbitrary, even negative values).

* void f1()-Do not edit this method. Your task is to complete the addLast(...) method above only. Output in the file f1.txt must be the following: (B,5,3) (C,6,5) (D,2,4) (E,7,9) (F,4,7)
* void f2()-There are 2 given Castor objects x, y in this function. Suppose the list contains at least 3 elements. Write statements to insert x and y to the list so that y will be the 1st (head), x will be the 4th element in the list. Output in the file f2.txt must be the following: (C,9,8) (D,6,3) (E,8,5) (F,5,4) (I,4,9)

(Y,3,4) (C,9,8) (D,6,3) (X,1,2) (E,8,5) (F,5,4) (I,4,9)

* void f3() - Suppose the list is not empty. Find the (first) node having maximum type and change its' place to YY. Output in the file f3.txt must be the following: (C,8,6) (D,3,5) (E,9,2) (F,5,8) (G,9,7) (H,6,8) (I,7,3)

## Question 2: (4 marks)

In this question you should complete some methods in BSTree.java file.

The class Castor with 3 data members: place, depth and type is given and you do not need to edit it. The BSTree class is a binary search tree of Castor objects. The variable depth is the key of the tree, thus it must be unique. The following methods should be completed:

void insert(String xPlace, int xDepth, int xType)-check if xPlace.charAt(0)=='B' then do nothing, otherwise insert new Castor with place=xPlace, depth-xDepth, type-xType to the tree (depth and type can get arbitrary, even negative values).

* void f1()- Do not edit this method. Your task is to complete the insert(...) method above only. Output in the file f1.txt must be the following: (A,7,9) (C,4,3) (D,8,6) (E,2,5) (F,6,7)

(E,2,5) (C,4,3) (F,6,7) (A,7,9) (D,8,6)

* void f2()-Perform post-order traversal from the root but display to file f2.txt nodes with type<7 only. Hint: Copy the function postOrder(...) to function postOrder2(...) and modify it. Output in the file f2.txt must be the following: (E,2,8) (I,5,4) (H,6,3) (F,4,5) (G,8,7) (D,7,2) (C,3,6)

(I,5,4) (H,6,3) (F,4,5) (D,7,2) (C,3,6)

* void f3() -Perform breadth-first traversal and find the second node p having left son. Delete the node having largest depth in the subtree with root p (thus if p is the leaf-node then p is deleted). Output in the file f3.txt must be the following:

(C,5,2) (D,2,1) (E,6,5) (F,1,3) (G,4,6) (H,3,4)

(C,5,2) (D,2,1) (E,6,5) (F,1,3) (H,3,4)

-- lười chưa làm bằng cách mới

* void f4()-Perform breadth-first traversal and find the second node p having left son, then rotate p to right. Output in the file f4.txt must be the following: (C,5,2) (D,2,1) (E,6,5) (F,1,3) (G,4,6) (H,3,4)

(C,5,2) (F,1,3) (E,6,5) (D,2,1) (G,4,6) (H,3,4)

## Question 3: (2 marks)

In this question you should complete some methods in Graph.java file.

The class Graph is the implementation of a graph. The following methods should be completed:

* void f1()-Perform **depth-first** traversal (to the file f1.txt) from the vertex i=0 (the vertex A) but display 4 vertices from the 2nd vertex to the 5th vertex only. Hint: copy depth(...) to depth2(...) and modify the latter one. Content of the output file f1.txt must be:

A B E H I C D G F

B E H I

* void f2() - Apply the Dijkstra's shortest path algorithm to find (1) the shortest path from vertext 1 (B) to vertex 5 (F), then (2) from vertex 0 (A) to vertex 6 (G). Write 3 lines to the file f2.txt: line 1 contains vertices in shortest path (1), line 2 contains vertices in shortest path (2), t line 3 contains the last 3 vertices selected into the set S with their labels in (2). (Note thain the weighted matrix, the value 99 is considered as infinity). Output in the file f2.txt must be the following:

BCED F

ABC EDG

D-19 F-24 G-29

# TRIAL – SU23 ==DONE

## Question 1: (4 marks)

(Do not pay attention to real meaning of objects, variables and their values in the questions below). In this question you should complete some methods in MyList.java file.

The class Car with 2 data members: owner and price is given and you do not need to edit it. The MyList class is a linked list of Car objects. The following methods should be completed:

void addLast(String xOwner, int xPrice) - check if **xOwner.charAt(0) = 'B'** or **xPrice>100** then do nothing, otherwise add new car with owner-xOwner, price=xPrice to the end of the list. (price can get arbitrary value, even negative).

* void f1() This method is used to test the addLast methode above. You do not need to edit this function. Output in the file fl.txt must be the following:

(A.9) (C,7) (D.2) (E.6) (F.4)

* void f2() - There is a given objects x. You should write statements so that x will be the first element of the list. Output in the file f2.txt must be the following:

(C,9) (D.6) (E.8) (F.2) (I,6)

(X.1) (C.9) (D.6) (E,8) (F.2) (I,6)

* void f3() Suppose the list contains at least 3 elements. Delete the **first node having price 5**. Output in the file f3.txt must be the following:

(C.9) (D,5) (E.3) (F.5) (I,6)

(C.9) (E.3) (F.5) (I,6)

- void f4() - Sort the list ascendingly by price. Output in the file f4.txt must be the following:

(C,9) (D,2) (E,5) (F,13) (I,6) (J,1)

(J,1) (D,2) (E,5) (I,6) (C,9) (F,13)

## Question 2: (4 marks)

In this question you should complete some methods in BSTree.java files.

The class Car with 2 data members: owner and price is given and you do not need to edit it. The BSTree class is a binary search tree of Car objects. The variable **price** is the key of the tree. The following methods should be completed:

void insert(string xOwner, int xPrice) - check if **xOwner.charAt(0) = 'B' or xPrice 100** then do nothing, otherwise insert new car with owner=xOwner, price=xPrice to the tree.

* void f1() You do not need to edit this function. Your task is to complete the insert(...) function above only. Output in the file fl.txt must be the following:

(A,5) (C,2) (E,4) (G,3) (D,6) (F,7) (Y,105)

(C,2) (G,3) (E,4) (A,5) (D,6) (F,7) (Y,105)

* void f2() - Perform pre-order traversal from the root but display to file f2.txt nodes having price in the interval **[3,5] only**. Hint: Copy the function preOrder(...) to preOrder2(...) and modify it. Output in the file f2.txt must be the following:

(C,6) (D,2) (F,4) (H,3) (I,5) (E,8) (G,7)

(F,4) (H,3) (I,5)

* void f3() Perform **breadth-first** traversal from the root and **delete by copying** the first node having both 2 sons and price <7. Output in the file f3.txt must be the following:

(C,8) (D,6) (E,9) (F,2) (G,7) (H,1) (I,3) (J,5) (K,4)

(C,8) (J,5) (E,9) (F,2) (G,7) (H,1) (I,3) (K,4)

* void f4() Perform breadth-first traversal from the root and find the first node p having **left son** and **price <7**. Rotate p to **right about its left son**. Output in the file f4.txt must be the following:

(C,8) (D,6) (E,9) (F,2) (G,7) (H,1) (I,3) (J,5) (K,4)

(C,8) (F,2) (E,9) (H,1) (D,6) (I,3) (G,7) (J,5) (K,4)

## Question 3: (2 marks)

In this question you should complete some methods in Graph.java file.

The class Graph is the implementation of a graph. The following methods should be completed:

* void f1() - Perfom depth-first traversal (to the file fl.xt) from the **vertex i=1** (the vertex B) but display vertices with their **deegrees** in bracket. Hint: copy depth(...) to depth2(...) and modify the latter one. Content of the output file fl.txt must be:

B G A E F I C H D

B(1) G(2) A(4) E(3) F(3) I(3) C(1) H(2) D(1)

* void f2() - Apply the Dijkstra's shortest path algorithm to find the shortest path from the vertex **0 (A) to the vertex 4 (E).** (Note that in the weighted matrix, the value 999 is considered as infinity). Write 2 lines into the file f6.txt. The first line contains the list of **vertices in the shortest path**. The second lines contains **shortest distances to the vertices** in the first line. Content of the output file f2.txt must be:

AC FE

0 9 11 20

void f3() Supposed the given graph has Euler's cycle. Apply the pseudocode in the Graph.java file to write statements to find the Euler's cycle **from the vertex 1 (B).** Output in the file f3.txt must be the following:

B D E D C B E G F A B

# CAU 2

## F1- insert vao tree

void insert(String xSea, int xSail, int xPaddle) {

if (xSea.charAt(0)== 'B') return;

insert(new Boat(xSea, xSail, xPaddle));

}

void insert(Boat x) {

Node q = new Node(x);

if(isEmpty()) {root=q;return;}

Node f, p; f=null; p=root;

while(p!=null) {

if(p.info.sail==x.sail) {

return;

}

f=p;

if(x.sail<p.info.sail) p=p.left; else p=p.right;

}

if(x.sail<f.info.sail) f.left=q; else f.right=q;

}

## F2 - modify cac ham co san

//vi du

void preOrder2(Node p, RandomAccessFile f) throws Exception {

if(p==null) return;

if (p.info.color<6){

fvisit(p,f);

}

preOrder2(p.left,f);

preOrder2(p.right,f);

}

public void breadth2(Node p, RandomAccessFile f) throws Exception {

if(p == null)

return;

Queue q = new Queue();

q.enqueue(p);

Node r;

while(!q.isEmpty()) {

r = q.dequeue();

if(r.info.color<6){

fvisit(r,f);

}

if(r.left != null)

q.enqueue(r.left);

if(r.right != null)

q.enqueue(r.right);

}

}

## F3 - tim so cay con

### //tim node thu 5 (node3 luu gia tri node can tim)

int count3 = 0;

Node node3 = null;

void postOrder3(Node p) {

if (p == null) {

return;

}

postOrder3(p.left);

postOrder3(p.right);

//logic

count3++;

if (count3 == 5) {

node3 = p;

return;

}

}

### //tim so cay con

int countNode(Node pNode) {

if (pNode == null) {

return 0;

}

int k, h, rNode;

k = countNode(pNode.left);

h = countNode(pNode.right);

rNode = k + h + 1;

return rNode;

}

## delete by copying

//delete by copying

|  |
| --- |
| public void deleteCopying(int x) { //Delete Node with info = x  if (isEmpty()) {  return;  }  Node p = root;  Node f = null;  while (p != null) {  if (p.info.hoof == x) {  break;  }  f = p;  if (x < p.info.hoof) {  p = p.left;  } else {  p = p.right;  }  } //end while  if (p == null) {  return;  }  // p has no children  if (p.left == null && p.right == null) {  if (f == null) {  root = null;  return;  }  if (f.left == p) {  f.left = null;  return;  } else {  f.right = null;  return;  }  }  // p has only left child  if (p.left != null && p.right == null) {  if (f == null) {  root = root.left;  return;  }  if (f.left == p) {  f.left = p.left;  return;  } else {  f.right = p.left;  return;  }  }  // p has only right child  if (p.left == null && p.right != null) {  if (f == null) {  root = root.right;  return;  }  if (f.left == p) {  f.left = p.right;  return;  } else {  f.right = p.right;  return;  }  }  // p has 2 children  if (p.left != null && p.right != null) {  Node q = p.left;  Node rp = q;  Node fr = null;  while (rp.right != null) {  fr = rp;  rp = rp.right;  } //end while;  p.info = rp.info;  if (fr == null) {  p.left = q.left;  } else {  fr.right = rp.left;  }  return;  }  } |

## F4 - tim node thu may voi yeu cau cu the

### //tim node thu 2 co 2 con voi breath (node3 luu gia tri node can tim)

int count3=0;

Node node3 = null;

void breadth3(Node p, RandomAccessFile f) throws Exception {

if(p==null) return;

Queue q = new Queue();

q.enqueue(p);Node r;

while(!q.isEmpty()) {

r = q.dequeue();

if(r.left!=null && r.right!=null)

count3++;

if(count3==2){

node3=r;

return;

}

if(r.left!=null) q.enqueue(r.left);

if(r.right!=null) q.enqueue(r.right);

}

}

### // tim node thu 2 co cay con trai voi postOrder (node4 luu gia tri node can tim)

int count4 = 0;

Node node4 = null;

void postOrder4(Node p) {

if (p == null) {

return;

}

postOrder4(p.left);

postOrder4(p.right);

//logic

if (p.left != null){

count4++;

if (count4 == 2) {

node4 = p;

return;

}

}

}

## cac ham de quay

### //quay phai

void rotateRight(Node p, Node fp) {

if (p == null || p.left == null) {

return ;

}

Node p1 = p.left;

p.left = p1.right;

p1.right = p;

if (fp.left == p)

fp.left = p1;

else

fp.right = p1;

}

### //quay trai

void rotateLeft(Node p, Node fp) {

if (p == null || p.right == null) {

return ;

}

Node p1 = p.right;

p.right = p1.left;

p1.left = p;

if (fp.right == p)

fp.right = p1;

else

fp.left = p1;

}

### //tim parent

Node searchParent(Node a) {

if (a == null)

return null;

Node p = root, f = null;

while (p != null && p != a) {

f = p;

if (p.info.weight > a.info.weight)

p = p.left;

else

p = p.right;

}

return f;

}

## tinh height cua mot node

public int height(Node p) {

if (p == null) {

return 0;

}

int l = height(p.left) + 1;

int r = height(p.right) + 1;

return (l > r) ? l : r;

}

## tim node co gia tri max

//tim max (node cuối theo cách preOder là max)

int count4=0;

Node node4 = null;

void preOrder4(Node p, RandomAccessFile f) throws Exception {

if(p==null) return;

int size = this.countNode(root);

count4++;

if(count4==size){

node4=p;

return;

}

preOrder4(p.left,f);

preOrder4(p.right,f);

}

int countNode(Node pNode) {

if (pNode == null) {

return 0;

}

int k, h, rNode;

k = countNode(pNode.left);

h = countNode(pNode.right);

rNode = k + h + 1;

return rNode;

}

// ham main

this.preOrder4(root, f);

Node fp=this.searchParent(node4);

if(fp!=null){

Node ffp=this.searchParent(fp);

if(fp.left==node4)

this.rotateRight(fp, ffp);

else{

this.rotateLeft(fp, ffp);

}

}

## tim node co gia tri min

//tim min (node đầu theo cách postOrder la min)

int count4=0;

Node node4 = null;

void postOrder4(Node p, RandomAccessFile f) throws Exception {

if(p==null) return;

postOrder4(p.left,f);

postOrder4(p.right,f);

count4++;

if(count4==1){

node4=p;

return;

}

}

//ham main

postOrder4(root, f);

Node fp= searchParent(node4);

if (fp != null){

Node fpp= searchParent(fp);

if (fp.left == node4)

this.rotateRight(fp, fpp);

else

this.rotateLeft(fp, fpp);

}

*The****depth****of a node is the number of edges present in path from the root node of a****tree****to that node.  
The****height****of a node is the number of edges present in the longest path connecting that node to a leaf node.*

*Lá:*

*depth: độ dài cả cây*

*Height: 0*