Question 2: (5 marks)

Write a program in Java using Binary Search Tree data structure to manage information about persons. Variables used to store information about a person are:

- name - the name of a person (character String) , which is the **key of the tree**.
- age - the age of a person (integer value).

You should write the **BSTree** class, which is a **binary search tree** data structure to store person information. Behind some other functions, the following functions should be included in the BSTree class:

- void insert(String xName, int xAge) - check if **the first letter of xName is 'B'** (i.e. xName.charAt(0) == 'B') then **do nothing**, otherwise insert new person with name=xName, age=xAge to the tree.
- void preOrder(Node p, RandomAccessFile f) - save all elements of the tree in format (name, age) to the file f by pre-order traverse. ***This function is given***.

- void f1() – Test insert function. You do not need to edit this function. Your task is to complete the insert(String xName, int xAge) function only.
  With the given data, the content of the file **f1.txt must be the following:**:
  (A6,1) (A2,2) (A1,4) (A5,5) (A4,7) (A3,9) (A7,3) (A9,6) (A8,8)
  (A1,4) (A2,2) (A3,9) (A4,7) (A5,5) (A6,1) (A7,3) (A8,8) (A9,6)

- void f2() – create BSTree object h and using insert method to insert to h all elements having age>4. You should visit elements by breadth-first traverse
  With the given data, the content of the file **f2.txt must be the following:**:
  (C6,1) (C2,2) (C1,4) (C5,5) (C4,7) (C3,9) (C7,3) (C9,6) (C8,8)
  (C5,5) (C4,7) (C3,9) (C9,6) (C8,8)

- void f3() – calculate the height of the tree. You should use the given statement f123.writeBytes("k = " + k + "\r\n"); to write this value to the file f3.txt.
  With the given data, the content of the file **f3.txt must be the following:**
  k = 5

- void f4() - calculate the number of nodes of the tree. You should use the given statement f123.writeBytes(" k = " + k + "\r\n"); to write this value to the file f4.txt.
  With the given data, the content of the file **f4.txt must be the following:**
  k = 9

- void f5() – Delete the root of the tree by copying.
  With the given data, the content of the file **f5.txt must be the following:**
  (C6,1) (C2,2) (C1,4) (C5,5) (C4,7) (C3,9) (C7,3) (C9,6) (C8,8)
  (C5,5) (C2,2) (C1,4) (C4,7) (C3,9) (C7,3) (C9,6) (C8,8)

- void f6() – Check if the root having non-empty left-son then rotate it to right about its left-son.With the given data, the content of the file **f6.txt must be the following:**
  (C6,1) (C2,2) (C7,3) (C1,4) (C5,5) (C9,6) (C4,7) (C8,8) (C3,9)
  (C2,2) (C1,4) (C6,1) (C5,5) (C7,3) (C4,7) (C9,6) (C3,9) (C8,8)

- void f7() – Calculate balance factor of all nodes. Display all node with balance factor by breadth-first traverse. Display also the information about whether a given binary search tree is height balanced (AVL tree) or not.
  With the given data, the content of the file **f7.txt must be the following:**
  (C6,1) (C2,2) (C7,3) (C1,4) (C5,5) (C9,6) (C4,7) (C8,8) (C3,9)
  (C6,1,-1) (C2,2,2) (C7,3,2) (C1,4,0) (C5,5,-2) (C9,6,-1) (C4,7,-1) (C8,8,0) (C3,9,0)
  The tree is not an AVL tree

- void f8() – Calculate level of all nodes. Display all node with level by breadth-first traverse.
  With the given data, the content of the file **f8.txt must be the following:**

(C6,1) (C2,2) (C7,3) (C1,4) (C5,5) (C9,6) (C4,7) (C8,8) (C3,9)
(C6,1,1) (C2,2,2) (C7,3,2) (C1,4,3) (C5,5,3) (C9,6,3) (C4,7,4) (C8,8,4) (C3,9,5)

- void f9() – Balance a binary search tree by simple balancing algorithm. Display all node by breadth-first traverse.
  With the given data, the content of the file **f9.txt must be the following:**
  (C6,1) (C2,2) (C7,3) (C1,4) (C5,5) (C9,6) (C4,7) (C8,8) (C3,9)
  (C5,5) (C2,2) (C7,3) (C1,4) (C3,9) (C6,1) (C8,8) (C4,7) (C9,6)