Question 2. (15 marks)
Write a new method for the binary search tree class (BST) that takes two keys, *low* and *high*, and prints all elements X that are in the range specified by *low* and *high*.

void printBetween(int low, int high)

Solution

```
void Print(BSTNode<T> t, int k1, int k2)
{
     if (t == null)
          return;

     if (k1 < t.data)
          Print(t.left, k1, k2);

     if (k1 <= t.data && k2 >= t.data)
          System.out.println(t.data);

     if (k2 > t.data)
          Print(t.right, k1, k2);
}
```

Question 3. (20 marks)

(a) Using the specification of Stack ADT, implement the following operations.
RetriveBottom (Stack s; Type e)
Precondition/Requires: The stack s is non-empty.
Results/Actions: Returns only the data value of the data element at the bottom of stack s, leaving the stack unchanged.

(b) Using specification of Queue ADT, implement the following operation. You may use another data structure.
ReverseQueue (Queue q)
Precondition/Requires: The queue q is non-empty.
Results/Actions: Reverses the order of elements in queue q.

Question 1. (25 marks)

(a) Translate the following infix expression into postfix expression:
(a / (b / c)) * (d – e).
(b) Give one advantage of representing arithmetic expressions in postfix form.
(c) Show the state of the array, the values of head and tail in the following array- based queue (with wrap-round), after the following statements are executed.

```
ArrayQueue<int> q = new ArrayQueue<int>(5);
for (int i = 1;i <= 5;i++) q.enqueue(i);
q.serve();
q.serve();
```

```
q.enqueue(7);
```

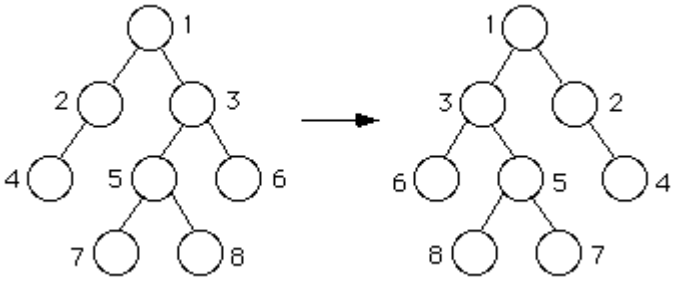
(d) Write a static method to remove the last element in a queue without using any other data structure.

```
public static <int> void removeLast (Queue<int> q);
```

Question 2. (30 marks)

(a) Implement a new method in the ***BinaryTree<int>*** class to compute the average of all the nodes in the tree.

(b) Add a new method to the BinaryTree class: ***void mirror(BTNode<T> t)***, that uses recursion to interchange left and right subtrees of a all nodes. The method is called with parameter t as the root of the tree.



Question 3 (15 marks)

Given the following method, draw the calling tree for the call ***g(1,3)***

```
public static int g(int a, int b){
     if (a == 0 || b == 0)
          return a-b;
     else
          return g(a, b-1) + g(a-1, b);
}
```

Question 4 (30 marks)

(a) Insert the following keys into an empty BST and show the BST after every insert. The keys are: 17, 10, 20, 13, 9, 25, 18, 19.

(b) Now delete 17 from the BST you obtained in part (a).

Question 2 (15)

Implement a static method in the test class for ADT Priority Queue to merge two priority queues. Use the operations in the specification of ADT Priority Queue.

```
public static<T> Stack<T> concatenate(Stack<T> s1, Stack<T> s2)
```

Question 3

[25 points]

1. Write a static method (User of ADT) named replace that accepts a queue q and two integers i and j, and replaces the element at position i with the one at position j (the first element in the queue has position 0). Element j is not modified. The queue order should not change otherwise. Assume that $0 \leq i < n$ and $0 \leq j < n$, where n is the length of the queue (notice that we may have $i \geq j$ as well as $i \leq j$). The method signature is static <T >void replace(Queue <T >q, int i, int j).

Example 3.1. If q contains: A → B → C → D → E, then after the call replace(q,1,4), q becomes: A → E → C → D → E.

2. Rewrite the previous method as a member of the class LinkedQueue.

3. Give the big-oh notations for the running time of the previous two methods and compare them.

Question 3

[25 points]

1. Write a static method findSmallest (user of the ADT queue) that takes as input a non-empty queue q of integers and finds the smallest number in this queue (q must not change). The method signature is: public static Integer findSmallest(Queue <Integer >q). You can compare between variables of type Integer using the usual operators: ==, <, >, etc.

Example 3.1. If q : 2 → 4 → 3 → 8 → 3 → 6 → 1 → 9, then findSmallest(q) returns ”1”, as it is the smallest number in the queue.

2. Write the static method exchange (user of the ADT stack) that takes as input two stacks st1 , st2 and exchanges their contents. The method signature is: public static <T >void exchange (Stack <T >st1 , Stack<T >st2).