Lab 7

Houssam Eddine ATIF

N# 610165

1/

```
public static String reverse(String a) {

		Stack<Character> st=new Stack<Character>();
		String r="";

		int i=0;

		while(i<a.length()) {
			while(i!=a.length() && a.charAt(i)!=' ') {
				st.push(a.charAt(i));
				i++;
			}

			while(!st.empty()){
				r=r+(Character)st.pop();
			}
			i++;
			r=r+" ";
		}


		return r;
	}
```

2/

```
@Override
	public int[] sort(int[] arr) {
		// TODO Auto-generated method stub
	  MyBST t=new MyBST();
	  List<Integer> l=new ArrayList<>();

		for(int i=0;i<arr.length;i++) {
			t.insert(new Integer(arr[i]));
		}

		helper(t.root,l);

		for(int i=0;i<arr.length;i++) {
			arr[i]=l.get(i);
		}

		return arr;
	}

	private void helper(Node t, List l) {
		if (t != null) {
			helper(t.left,l);
			l.add(t.element);
			helper(t.right,l);
		}
	}
```

In this algorithm we insert the array in the BST, in the average case Insertion operation of tree take O(nLog(n)), after that we traverse the tree and add each element to a list, which takes O(nlog(n)) and finally we have a loop that copy the List to an array and return the array witch takes O(n) the sum is 2*O(nlog(n))+O(n)=O(nlog(n)).

The result for sorting test is:

```
8 ms -> MergeSort
34 ms -> MyBST
42 ms -> MergeSortPlus
```

Witch is a good result for BST sorting method it's near to the fast sorting method, sorting with BST method is not always faster because here we are talking about the average case in the worst case the BST will behave like a linked list.

3/

Num nodes n	Does there exist a red-black tree with n nodes, all of which are black?
1	Yes
2	No
3	Yes
4	No
5	No
6	No
7	Yes

4/

Num nodes n	Does there exist a red-black tree with n nodes, where exactly one of the nodes is red?
1	No
2	Yes
3	No
4	Yes
5	Yes
6	No
7	No