**SortStack**

**Q:** Given a stack, sort it using recursion. Use of any loop constructs like while, for..etc is

not allowed. We can only use the following ADT functions on Stack S:

is\_empty(S) : Tests whether stack is empty or not.

push(S) : Adds new element to the stack.

pop(S) : Removes top element from the stack.

top(S) : Returns value of the top element. Note that this

function does not remove element from the stack.

**Test Cases:**

**Input:** 7 3 8 1

**Output:** 8 7 3 1

**Input:** -3 14 18 -5 30

**Output:** 30 18 14 -3 -5

**Input:** 8 14 -34 0 2

**Output:** 14 8 2 0 -34

**Algorithm:**

Step-1: Push all the elements in the stack.

Step-2: Recursively keep popping out the values and hold them.

Here is the sample code for the above step:

void popAll( ) {

if the stack is not empty {

temp = pop( );

popAll( ); //recursive calling of the same function

insertInOrder(temp); //calling the function to insert the values in order

}

}

Step-3: insert them in the stack in sorted order.

Here is the sample code for the above step:

void insertInOrder(element) {

if stack is empty

push(element);

else {

if(peek( ) >= element)

push(element);

else {

temp = pop( );

insertInOrder(element);

push(temp);

}

}

}

**Example:**

Input: 7 3 8 1 (Elements pushed on the stack.) 7 on the top of the stack.

* recursively all the values are popped from the stack.
* Insert them in order back to the stack.
* Insert 7 first.
* Since 7 > 3, hence push 3.
* 8 > 7, pop 7 and 3 as well recurively, push 8, push 7, push 3.
* 8 > 1, push 1 to the stack and hence will come to the bottom of the stack.