

STACK

I. Java Implementation

Ia. Array based implementation

push

```
@Assertion(value = { @Case(
pre = "theArray!=null && theArray.length>0 && !isFull() && topOfStack>=-1 && topOfStack <
    theArray.length",
post = "theArray[topOfStack]==x" ) })
public void push( Object x ) throws Overflow
{
    if( isFull( ) )
        throw new Overflow( );
    theArray[ ++topOfStack ] = x;
}
```

pushPop

```
@Assertion(value = { @Case(
pre = "theArray!=null && theArray.length>0 && !isFull() && topOfStack>=-1 && topOfStack <
    theArray.length",
post = "true" ) })
public void pushPop(Object x) throws Overflow{
    push(x);
    Object y = topAndPop();
    assert x == y;
}
```

isFull

```
public boolean isFull( )
{
    return topOfStack == theArray.length - 1;
}
```

Iib. List Based Implementation

push

```
@Assertion(value = { @Case(
pre = "isAcyclic()",
post = "isAcyclic() && topOfStack.element==x" ) })
public void push( Object x )
{
```

```
topOfStack = new ListNode( x, topOfStack );  
}
```

isAcyclic

```
public boolean isAcyclic()  
{  
    StackLi ll = new StackLi();  
    ListNode temp = topOfStack;  
    while (temp != null)  
    {  
        if (ll.contains(temp))  
        {  
            return false;  
        }  
        ll.topOfStack=new ListNode(temp,ll.topOfStack);  
        temp = temp.next;  
    }  
    return true;  
}
```