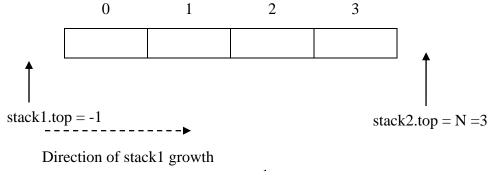
1. Explainhow to implement(or) write the algorithm two stacks in one array A[1 . . n] in such a way that neither stack overflows unless the total number of elements in both stacks together is n. The PUSH and POP operations should run in O(1) time.

## Key:

Let N refers the size of the Array



Direction of stack2 growth

```
Algorithm push_stack2(e)
if(stack1.size()+stack2.size() < N) then
    A[--stack2_top]=e
else
    print "StackOverflow"
```

```
Algorithm pop_stack1()

if (stack1.isEmpty()) then

print "Stack Empty Exception"

return null

else
```

```
temp=A[stack1_top]
A[stack1_top]=null
stack1_top= stack1_top -1
return temp
```

```
Algorithm pop_stack2()
  if (stack2.isEmpty()) then
     print "Stack Empty Exception"
     return null
  else

     temp=A[stack2_top]
     A[stack2_top]=null
     stack2_top = stack2_top+1
     return temp

Algorithm size_stack1()
    return stack1.top+1

Algorithm size_stack2()
    return N-stack2.top
```

2. In HTML, tags exist in both opening and closing forms and must be balanced to properly describe a web document. This very simple HTML document:

```
<html>
<head>
<title>
Example
</title>
</head>
<body>
<h1>Hello, world</h1>
</body>
</html>
```

Write an algorithm that can check an HTML document for proper opening and closing tags.

3. Write a recursive function to reverse a list.

Key:

```
Algorithm reverse ( a[],s, e)

if (s>=e) then

return

else

temp=a[s];
a[s]=a[e];
a[e]=temp;
reverse(a,s+1,e-1);
end
```

4. Show recursive trace for the function call symmetric and 10 40 0 20 30 are the inputs. **Algorithm** symmetric()

Input: n and m are inputs taken from the user // read the first integer Output: Returns True if reaches middle of the sequence or first input and last input matches. Otherwise, returns False

```
n=readNextInput()

if (n == 0) then

return True // we are in the middle of the sequence

else

// read the sequence in the middle and check whether it is symmetric

booleansim = symmetric()

sym = readNextInput() // read the last integer

return (n == m) && sym;
```

## Key:

