Rotate an array of n elements to the right by k steps.

For example, with n = 7 and k = 3, the array [1,2,3,4,5,6,7] is rotated to [5,6,7,1,2,3,4]. How many different ways do you know to solve this problem?

public static void RotateArray(int[]arr,int k)

{

for (int i = 1; i <= arr.Length; i++)

{

arr[i-1] = i;

}

if (k > arr.Length)

k = k % arr.Length;

int index = 0;

int shift = k;

for (int i = 0; i < arr.Length; i++)

{

if (shift > arr.Length-1)

{

shift = shift - arr.Length ;

}

int val = arr[shift];

arr[shift] = arr[index];

arr[index] = val;

shift += k; ;

}

}

Given an input string, reverse the string word by word. A word is defined as a sequence of non-space characters.

The input string does not contain leading or trailing spaces and the words are always separated by a single space.

For example,  
Given s = "the sky is blue",  
return "blue is sky the".

public static string ReverseWords(string str)

{

string[] words = str.Split(' ');

StringBuilder sb = new StringBuilder();

int i = 0;

int j = words.Length-1;

while (i<j)

{

sb.Append(words[j]);

words[j] = words[i];

words[i] = sb.ToString();

sb.Clear();

i++;

j--;

}

for (int it = 0; it < words.Length; it++)

{

sb.Append(words[it]+" ");

}

return sb.ToString();

}

[LeetCode – Evaluate Reverse Polish Notation](http://www.programcreek.com/2012/12/leetcode-evaluate-reverse-polish-notation/)

Evaluate the value of an arithmetic expression in Reverse Polish Notation. Valid operators are +, -, \*, /. Each operand may be an integer or another expression. For example:

["2", "1", "+", "3", "\*"] -> ((2 + 1) \* 3) -> 9

["4", "13", "5", "/", "+"] -> (4 + (13 / 5)) -> 6

1. Naive Approach

This problem can be solved by using a stack. We can loop through each element in the given array. When it is a number, push it to the stack. When it is an operator, pop two numbers from the stack, do the calculation, and push back the result.

static void Main(string[] args)

{

String[] tokens = new String[] { "3", "3", "+","5","12", "-","-"};

Stopwatch st = new Stopwatch();

st.Start();

Solve(tokens);

st.Stop();

}

public static void Solve(String[] input)

{

Stack<int> stack = new Stack<int>();

int result = 0;

for (int i = 0; i < input.Length; i++)

{

int number;

if (int.TryParse(input[i], out number))

{

stack.Push(number);

}

else

{

if (input[i] == "+")

{

var item1 = stack.Pop();

var item2 = stack.Pop();

result = item1 + item2;

stack.Push(result);

}

if (input[i] == "\*")

{

var item1 = stack.Pop();

var item2 = stack.Pop();

result = item1 \* item2;

stack.Push(result);

}

if (input[i] == "-")

{

var item1 = stack.Pop();

var item2 = stack.Pop();

result = item1 - item2;

stack.Push(result);

}

}

}

Console.WriteLine(result);

}