DAY 8

1. A Barua number is a number which consists of only zeroes and ones and has only one 1.

Barua number will start with 1. Given numbers, find out the multiplication of the numbers.

Note: The input may contain one decimal number and all other Barua numbers. (Assume

that each number is very large and total number of values give is also very large)

Input 1: 100 10 12 1000

Output 1: 12000000

Input 2: 100 121 1000000000000000

Output 2: 12100000000000000000

Input 3: 10 100 1000

Output 3: 1000000

// C++ program to Find Number of digits

// in base b.

#include <iostream>

#include <math.h>

using namespace std;

// function to print number of

// digits

void findNumberOfDigits(long n, int base)

{

// Calculating log using base

// changing property and then

// taking it floor and then

// adding 1.

int dig = (int)(floor( log(n) /

log(base)) + 1);

// printing output

cout << "The Number of digits of "

<< "Number " << n << " in base "

<< base << " is " << dig;

}

// Driver method

int main()

{

// taking inputs

long n = 1446;

int base = 7;

// calling the method

findNumberOfDigits(n, base);

return 0;

}

// This code is contributed by Manish Shaw

// (manishshaw1)

2. Implement push, pop and find the minimum element in a stack in O(1) time complexity.

class Node {

constructor(val){

this.val = val;

this.next = null;

}

}

class ItemStack {

constructor(){

this.first = null;

this.last = null;

this.length = 0;

this.minStack = new MinStack();

}

push(val){

let newNode = new Node(val)

if (this.length === 0){

this.first = newNode;

this.last = newNode;

}else{

let temp = this.first;

this.first = newNode;

this.first.next= temp;

}

this.minStack.push(val)

this.length++;

return this.length;

}

pop(){

if (this.length === 0) return null;

let temp = this.first;

if (this.length === 1){

this.last = null;

}

this.first = this.first.next;

this.length--;

if(temp.val === this.minStack.first.val){

this.minStack.pop()

}

return temp;

}

min(){

if(this.length === 0) return undefined;

else return this.minStack.first.val;

}

}

class MinStack {

constructor(){

this.first = null;

this.last = null;

this.length = 0;

}

push(val) {

let newNode = new Node(val);

if (this.length === 0) {

this.first = newNode;

this.last = newNode;

this.length++;

}else {

if (val < this.first.val) {

let temp = this.first;

this.first = newNode;

this.first.next= temp;

this.length++;

}

}

return this.length;

}

pop() {

let temp = this.first;

if (this.length === 1){

this.last = null;

}

this.first = this.first.next;

this.length--;

return temp;

}

}

let itemStack = new ItemStack();

itemStack.push(3);

itemStack.push(4);

itemStack.push(1);

console.log(itemStack.min());

itemStack.pop();

console.log(itemStack.min());

itemStack.pop();

console.log(itemStack.min());

itemStack.pop();

console.log(itemStack.min());