Data structures and algorithms

Saravanakaruppu. K

Batch one

Question 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

Public static void main(String[] args)

{

Long binary1, binary2;

Int I = 0, remainder = 0;

Int[] sum = new int[100101210000];

Scanner in = new Scanner(System.in);

System.out.print(“Input first binary number: “);

Binary1 = in.nextLong();

System.out.print(“Input second binary number: “);

Binary2 = in.nextLong();

While (binary1 != 0 || binary2 != 0)

{

Sum[i++] = (int)((binary1 % 10 + binary2 % 10 + remainder) % 2);

Remainder = (int)((binary1 % 10 + binary2 % 10 + remainder) / 2);

Binary1 = binary1 / 10;

Binary2 = binary2 / 10;

}

If (remainder != 0) {

Sum[i++] = remainder;

}

--I;

System.out.print(“Sum of two binary numbers: “);

While (I >= 0) {

System.out.print(sum[i--]);

}

System.out.print(“\n”);

}

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

// C++ program to implement a stack that supports

// getMinimum() in O(1) time and O(1) extra space.

#include <bits/stdc++.h>

Using namespace std;

// A user defined stack that supports getMin() in

// addition to push() and pop()

Struct MyStack

{

Stack<int> s;

Int minEle;

// Prints minimum element of MyStack

Void getMin()

{

If (s.empty())

Cout << “Stack is empty\n”;

// variable minEle stores the minimum element

// in the stack.

Else

Cout <<”Minimum Element in the stack is: “

<< minEle << “\n”;

}

// Prints top element of MyStack

Void peek()

{

If (s.empty())

{

Cout << “Stack is empty “;

Return;

}

Int t = s.top(); // Top element.

Cout << “Top Most Element is: “;

// If t < minEle means minEle stores

// value of t.

(t < minEle)? Cout << minEle: cout << t;

}

// Remove the top element from MyStack

Void pop()

{

If (s.empty())

{

Cout << “Stack is empty\n”;

Return;

}

Cout << “Top Most Element Removed: “;

Int t = s.top();

s.pop();

// Minimum will change as the minimum element

// of the stack is being removed.

If (t < minEle)

{

Cout << minEle << “\n”;

minEle = 2\*minEle – t;

}

Else

Cout << t << “\n”;

}

// Removes top element from MyStack

Void push(int x)

{

// Insert new number into the stack

If (s.empty())

{

minEle = x;

s.push(x);

cout << “Number Inserted: “ << x << “\n”;

return;

}

// If new number is less than minEle

If (x < minEle)

{

s.push(2\*x – minEle);

minEle = x;

}

Else

s.push(x);

cout << “Number Inserted: “ << x << “\n”;

}

};

// Driver Code

Int main()

{

MyStack s;

s.push(3);

s.push(5);

s.getMin();

s.push(2);

s.push(1);

s.getMin();

s.pop();

s.getMin();

s.pop();

s.peek();

return 0;

}