CCC JUNIOR

Stack/Queue

今日课程预览

- Homework from Week 3
- Stack Concept
- Queue Concept
- Stack Exercises
- Queue Exercises
- 真题讲解

CCC 2005 J2 NSA Numbers

https://cemc.uwaterloo.ca/contests/computing/2005/stage1/juniorEn.pdf

- **Problem Description**

When a credit card number is sent through the Internet it must be protected so that other people cannot see it. Many web browsers use a protection based on "RSA Numbers."

A number is an RSA number if it has exactly four divisors. In other words, there are exactly four numbers that divide into it evenly. For example, 10 is an RSA number because it has exactly four divisors (1, 2, 5, 10). 12 is not an RSA number because it has too many divisors (1, 2, 3, 4, 6, 12). 11 is not an RSA number either. There is only one RSA number in the range 10...12.

Write a program that inputs a range of numbers and then counts how many numbers from that range are RSA numbers. You may assume that the numbers in the range are less than 1000.

Input and Output Specification

- **Program Output:** Enter lower limit of range

  User Input: 10

  Program Output: Enter upper limit of range

  User Input: 12

  Program Output: The number of RSA numbers between 10 and 12 is 1
- **Program Output:** Enter lower limit of range

  User Input: 11

  Program Output: Enter upper limit of range

  User Input: 15

  Program Output: The number of RSA numbers between 11 and 15 is 2

CCC 2013 J3 From 1987 to 2013

https://cemc.uwaterloo.ca/contests/computing/2013/stage1/juniorEn.pdf

- **Problem Description**

You might be surprised to know that 2013 is the first year since 1987 with distinct digits. The years 2014, 2015, 2016, 2017, 2018, 2019 each have distinct digits. 2012 does not have distinct digits, since the digit 2 is repeated. Given a year, what is the next year with distinct digits?

- **Input Specification**
  The input consists of one integer Y ($0 \leq Y \leq 10000$), representing the starting year.
- **Output Specification**

  The output will be the single integer D, which is the next year after Y with distinct digits.

Sample Input and Output

- Sample Input 1

  1987

- Output for Sample Input 1

  2013

- Sample Input 2

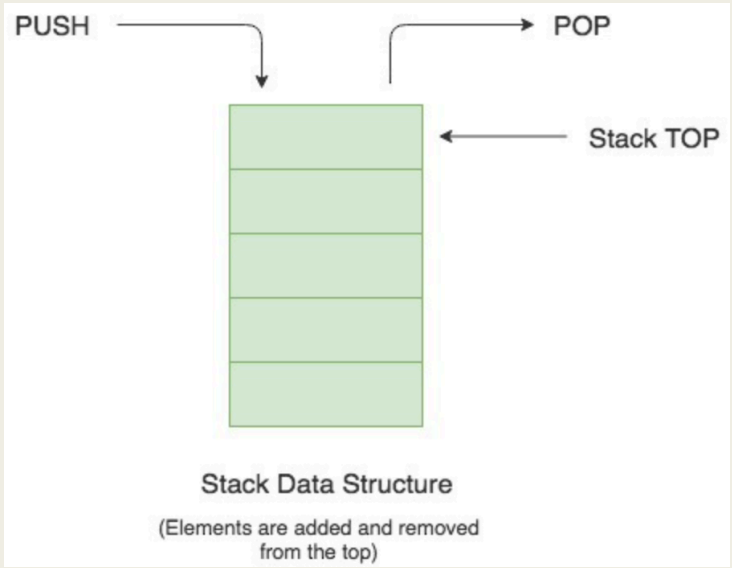
  999

- Output for Sample Input 2

  1023

Stack

- First In Last Out (FILO)



empty()	The method checks the stack is empty or not.
push(E item)	The method pushes (insert) an element onto the top of the stack.
pop()	The method removes an element from the top of the stack and returns the same element as the value of that function.
peek()	The method looks at the top element of the stack without removing it.
search(Object o)	The method searches the specified object and returns the position of the object.

Stack in Java

- import java.util.Stack;
- *Stack<E> stack = new Stack<E>();*
- *stack.push(element)*
- *stack.pop(element)*
- *stack.empty()*

Example of Stack - Valid Parentheses

- Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.An input string is valid if:
  - *Open brackets must be closed by the same type of brackets.*
  - *Open brackets must be closed in the correct order.*
- **Input:** s = "()" **Output:** true
- **Input:** s = "()[]{}" **Output:** true
- **Input:** s = "(]" **Output:** false
- **Input:** s = "([)]" **Output:** false
- **Input:** s = "{[]}" **Output:** true

Example of Stack – Baseball Game

- You are keeping score for a baseball game with strange rules. The game consists of several rounds, where the scores of past rounds may affect future rounds' scores.
- At the beginning of the game, you start with an empty record. You are given a list of strings ops, where ops[i] is the i^{th} operation you must apply to the record and is one of the following:
  - *An integer x - Record a new score of x.*
  - *"+" - Record a new score that is the sum of the previous two scores. It is guaranteed there will always be two previous scores.*
  - *"D" - Record a new score that is double the previous score. It is guaranteed there will always be a previous score.*
  - *"C" - Invalidate the previous score, removing it from the record. It is guaranteed there will always be a previous score.*
- Return *the sum of all the scores on the record*.

Sample Input/Output

- **Input:** ops = ["5","2","C","D","+"]

  Output: 30

  Explanation:

  "5" - Add 5 to the record, record is now [5].

  "2" - Add 2 to the record, record is now [5, 2].

  "C" - Invalidate and remove the previous score, record is now [5].

  "D" - Add 2 * 5 = 10 to the record, record is now [5, 10].

  "+" - Add 5 + 10 = 15 to the record, record is now [5, 10, 15]. The total sum is 5 + 10 + 15 = 30.

Sample Input/Output

- **Input:** ops = ["5","-2","4","C","D","9","+","+"]

  Output: 27

  Explanation:

  "5" - Add 5 to the record, record is now [5].

  "-2" - Add -2 to the record, record is now [5, -2].

  "4" - Add 4 to the record, record is now [5, -2, 4].

  "C" - Invalidate and remove the previous score, record is now [5, -2].

  "D" - Add 2 * -2 = -4 to the record, record is now [5, -2, -4].

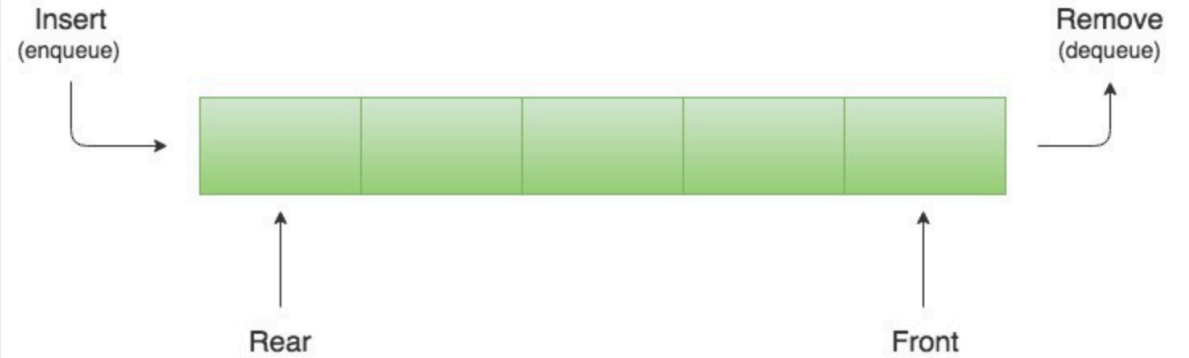
  "9" - Add 9 to the record, record is now [5, -2, -4, 9].

  "+" - Add -4 + 9 = 5 to the record, record is now [5, -2, -4, 9, 5].

  "+" - Add 9 + 5 = 14 to the record, record is now [5, -2, -4, 9, 5, 14]. The total sum is 5 + -2 + -4 + 9 + 5 + 14 = 27.

Queue

- First In First Out (FIFO)



Queue in Java

- import java.util.Queue;
- Queue<Integer> pQueue = new PriorityQueue<Integer>();
- Queue<Integer> ll = new LinkedList<Integer>();

boolean add(object)	It is used to insert the specified element into this queue and return true upon success.
boolean offer(object)	It is used to insert the specified element into this queue.
Object remove()	It is used to retrieves and removes the head of this queue.
Object poll()	It is used to retrieves and removes the head of this queue, or returns null if this queue is empty.
Object element()	It is used to retrieves, but does not remove, the head of this queue.
Object peek()	It is used to retrieves, but does not remove, the head of this queue, or returns null if this queue is empty.

Example of Queue – Topology Sort

- **Problem Description**

There are a total of numCourses courses you have to take, labeled from 0 to numCourses-1. Some courses may have prerequisites, for example to take course 0 you have to first take course 1, which is expressed as a pair: [0,1]

Given the total number of courses and a list of prerequisite **pairs**, is it possible for you to finish all courses?

Sample Input/Output

- **Input:** numCourses = 2, prerequisites = [[1,0]]

  Output: true

  Explanation: There are a total of 2 courses to take. To take course 1 you should have finished course 0. So it is possible.

- **Input:** numCourses = 2, prerequisites = [[1,0],[0,1]]

  Output: false

  Explanation: There are a total of 2 courses to take. To take course 1 you should have finished course 0, and to take course 0 you should also have finished course 1. So it is impossible.