//QUESTION1

package com.studyopedia;

import java.util.\*;

public class Java{

public static void main(String[]args) {

//QUESTION 1

int array[]= new int[5];

Scanner input= new Scanner(System.***in***);

System.***out***.print("enter array member1");

int member1 = input.nextInt();

System.***out***.print("enter array member2");

int member2 = input.nextInt();

System.***out***.print("enter array member3");

int member3 = input.nextInt();

System.***out***.print("enter array member4");

int member4 = input.nextInt();

System.***out***.print("enter array member5");

int member5 = input.nextInt();

array[0]=member1;

array[1]=member2;

array[2]=member3;

array[3]=member4;

array[4]=member5;

System.***out***.print("enter the number to search");

int search = input.nextInt();

for(int i:array) {

if(i==search)

System.***out***.println("it founds in the array");

}

int sum =0;

for(int j=0;j<array.length;j++) {

if (array[j]==search) {

sum=sum+1;

}}

System.***out***.print("the number founds "+sum+" times");

}

}

**QUESTION2**

package com.studyopedia;

import java.util.\*;

public class Rar {

int[] numArray;

int top;

int size;

public Rar(int capacity) {

numArray = new int[capacity];

top = -1;

size = 0;

}

public int size() {

return size;

}

void push(int value) {

if (top < numArray.length - 1) {

numArray[++top] = value;

size++;

}

else {

System.***out***.println("Full");

}

}

public int pop() {

if (top >= 0) {

int data = numArray[top--];

size--;

return data;

}

return -1;

}

public static void main(String[] args) {

Rar numStack = new Rar(5);

numStack.push(1);

numStack.push(2);

numStack.push(3);

numStack.push(4);

numStack.push(5);

numStack.pop();

numStack.pop();

for (int i = 0; i < numStack.size; i++) {

System.***out***.println(numStack.numArray[i]);

}

}

}

// QUESTION 3

package com.studyopedia;

import java.util.\*;

class Method {

private int maxSize;

private char[] stackArray;

private int top;

public Method(int size) {

maxSize = size;

stackArray = new char[maxSize];

top = -1;

}

public void push(char j) {

if (isFull()) {

System.***out***.println("Stack is full, cannot push " + j);

return;

}

stackArray[++top] = j;

}

public char pop() {

if (isEmpty()) {

System.***out***.println("Stack is empty");

return '\0';

}

return stackArray[top--];

}

public char peek() {

if (isEmpty()) {

System.***out***.println("Stack is empty");

return '\0';

}

return stackArray[top];

}

public boolean isEmpty() {

return (top == -1);

}

public boolean isFull() {

return (top == maxSize - 1);

}

}

class QueueUsingTwoStacks {

private Stack stack1;

private Stack stack2;

public QueueUsingTwoStacks(int capacity) {

stack1 = new Stack();

stack2 = new Stack();

}

public void enqueue(char item) {

stack1.push(item);

}

public Object dequeue() {

if (isEmpty()) {

System.***out***.println("Queue is empty");

return '\0';

}

if (stack2.isEmpty()) {

while (!stack1.isEmpty()) {

stack2.push(stack1.pop());

}

}

return stack2.pop();

}

public Object peek() {

if (isEmpty()) {

System.***out***.println("Queue is empty");

return "\0";

}

if (stack2.isEmpty()) {

while (!stack1.isEmpty()) {

stack2.push(stack1.pop());

}

}

return stack2.peek();

}

public boolean isEmpty() {

return (stack1.isEmpty() && stack2.isEmpty());

}

public static void main(String[] args) {

QueueUsingTwoStacks queue = new QueueUsingTwoStacks(5);

queue.enqueue('A');

queue.enqueue('B');

queue.enqueue('C');

System.***out***.println("Dequeue: " + queue.dequeue());

System.***out***.println("Peek: " + queue.peek());

queue.enqueue('D');

queue.enqueue('E');

while (!queue.isEmpty()) {

System.***out***.println("Dequeue: " + queue.dequeue());

}

}}

QUESTION4

public class Test {

public static void bubbleSort(char[] arr) {

int n = arr.length;

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (arr[j] > arr[j + 1]) {

// Swap arr[j] and arr[j+1]

char temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

public static void main(String[] args) {

char[] alphabets = {'d', 'a', 'f', 'b', 'c', 'e'};

*bubbleSort*(alphabets);

System.***out***.print("Sorted array: ");

for (char ch : alphabets) {

System.***out***.print(ch + " ");

}

}}

QUESTION 5

package com.studyopedia;

import java.util.\*;

public class discret{

public static void main(String[] args) {

int array[]= new int[5];

Scanner input= new Scanner(System.***in***);

System.***out***.print("enter array member1");

int member1 = input.nextInt();

System.***out***.print("enter array member2");

int member2 = input.nextInt();

System.***out***.print("enter array member3");

int member3 = input.nextInt();

System.***out***.print("enter array member4");

int member4 = input.nextInt();

System.***out***.print("enter array member5");

int member5 = input.nextInt();

array[0]=member1;

array[1]=member2;

array[2]=member3;

array[3]=member4;

array[4]=member5;

System.***out***.println("Original array: " + Arrays.*toString*(array));

*mergeSort*(array);

System.***out***.println("Sorted array: " + Arrays.*toString*(array));

}

private static void mergeSort(int[] inputArray) {

int inputLength = inputArray.length;

if (inputLength < 2) {

return;

}

int midIndex = inputLength / 2;

int[] leftHalf = new int[midIndex];

int[] rightHalf = new int[inputLength - midIndex];

for (int i = 0; i < midIndex; i++) {

leftHalf[i] = inputArray[i];

}

for (int i = midIndex; i < inputLength; i++) {

rightHalf[i - midIndex] = inputArray[i];

}

*mergeSort*(leftHalf);

*mergeSort*(rightHalf);

// Merge

*merge*(inputArray, leftHalf, rightHalf);

}

private static void merge(int[] inputArray, int[] leftH, int[] rightH) {

int leftSize = leftH.length;

int rightSize = rightH.length;

int i = 0, j = 0, k = 0;

while (i < leftSize && j < rightSize) {

if (leftH[i] <= rightH[j]) {

inputArray[k] = leftH[i];

i++;

} else {

inputArray[k] = rightH[j];

j++;

}

k++;

}

while (i < leftSize) {

inputArray[k] = leftH[i];

i++;

k++;

}

while (j < rightSize) {

inputArray[k] = rightH[j];

j++;

k++;

}

}}

QUESTION 6

package com.studyopedia;

import java.util.ArrayList;

import java.util.Scanner;

public class Test {

public static void main(String[] args) {

int[] array = {4, 6, 8, 9, 34};

int indexToDelete = 3;

int[] modifiedArray = deleteElement(array, indexToDelete);

for (int num : modifiedArray) {

System.out.print(num + " ");

}

}

public static int[] deleteElement(int[] array, int index) {

if (index < 0 || index >= array.length) {

System.out.println("Invalid index.");

return array; // Return the original array unchanged

}

// Create a new array with size one less than the original array

int[] newArray = new int[array.length - 1];

// Copy elements from the original array to the new array, skipping the element at the specified index

for (int i = 0, j = 0; i < array.length; i++) {

if (i != index) {

newArray[j++] = array[i];

}

}

return newArray;

}

}