**Problema 1**

Fie un sir de numere întregi cu 10 elemente. Sa se citească de la tastatura o poziție intre 0 – 9 si un număr care va fi introdus pe acea poziție.

Se va testa ca algoritmul funcționează corect daca se adaugă un număr pe **poziția 0**; **poziția 9** si **o poziție intermediara**.

Exemplu de sir [12, 25, 47, 58, 47, 7, 96, 14, 3, 9]

**Adăugare pe poziția 0 a elementului 100**

Rezultat: [**100**, 12, 25, 47, 58, 47, 7, 96, 14, 3, 9]

**Adăugare pe poziția 9 a elementului 100**

Rezultat: [12, 25, 47, 58, 47, 7, 96, 14, 3, 9, **100**]

**Adăugare pe poziția 4 a elementului 100**

Rezultat: [ 12, 25, 47, **100**, 58, 47, 7, 96, 14, 3, 9]

**!!** Sirul va fi de tipul **int[]** nu ArrayList<Integer>

* Reinițializat **int[]** cu un element in plus

**Problema 2**

Fie un sir de numere întregi cu 10 elemente. Sa se citească de la tastatura o poziție intre 0 – 9 iar algoritmul va șterge elementul de la poziția citita.

Se va testa ca algoritmul funcționează corect daca se adaugă un număr pe **poziția 0**; **poziția 9** si **o poziție intermediara**.

Exemplu de sir [12, 25, 47, 58, 47, 7, 96, 14, 3, 9]

**Șterge pe poziția 0**

Rezultat: [ 25, 47, 58, 47, 7, 96, 14, 3, 9, **null**]

**Șterge pe poziția 9**

Rezultat: [12, 25, 47, 58, 47, 7, 96, 14, 3, **null**]

**Șterge pe poziția 4**

Rezultat: [ 12, 25, 47, 47, 7, 96, 14, 3, 9, **null**]

**!!** Sirul va fi de tipul **int[]** nu ArrayList<Integer>

**Problema 3**

Fie un sir cu n elemente sortate crestator, se implementează o funcție cu doi parametri *(un sir de numere întregi, un număr ce vrea sa fie căutat)* care va trebui sa folosească **căutarea binară** pentru a găsii numărul dat ca si parametru in șirul primit ca si parametru. Funcția va returna **ture sau false**.

Se va testa algoritmul pentru următoarele cazuri:

* Se va testa cu un număr **din jumătatea din stânga a șirului**, odată **de pe o poziție para** si alta data **de pe o poziție impara**.
* Se va testa cu un **din jumătatea din dreapta a șirului**, odată **de pe o poziție para** si alta data **de pe o poziție impara**.
* Se va testa cu un **din mijlocul șirului**.
* Se va testa cu **primul număr din șir**.
* Se va testa cu **ultimul din șir**.
* Se va test cu un **număr care nu exista**.

Pentru a testa aceasta funcție se va apela succesiv pentru fiecare caz aceeași funcție cu parametri de intrare diferiți. (Șirul poate fi același in toate cazurile de testare). In consola se va returna pe cate o line un string care spune ce caz ca fost testa si care este rezultatul testului.

**In total trebuie sa fie 8 cazuri.**

**Problema 4**

Folosind interfața de Shape si clasele Rectangle, Square si Circle implementate in laboratorul 3. Sa se folosească operațiile polomorfice pe șiruri pentru a calcula cate culori din fiecare categorie exista.

Exemplu: presupunem ca avem inițializate mai multe shape-uri:

* Un Rect cu culoare #010101
* Un Square cu culoare #010101
* Un Circle cu culoare #022010
* Un Rect cu culoare #010321

Dupa parcurgerea polimorfica a sirului de Shape-uri trebuie sa se afiseze in consola urmatorul raport:

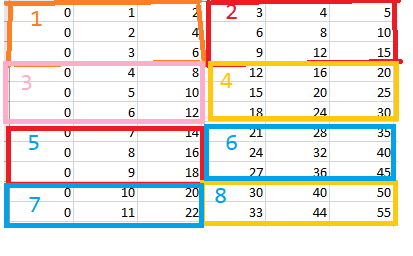
#010101 – 2

#022010 – 1

#010321 – 1

**Problema 5: Tablou bi dimensional:**   
Fie o imagine alb/negru (cu valori de la 0-255) stocata sub forma unui tablou bi dimensional cu n\*m pixel.  
0. Folosind **new Random().nextInt(255)** generate o imagine n\*m cu n si m citite de la tastatura  
1. Primul pas al programului este sa se afiseze in consola imaginea originala sub forma tablelara.  
  
255 2 150 5  
10 3 15 5   
14 2 4 5  
  
2. Pasul 2 – aplicarea unui filtru de 3\*3 care se “netezeasca” imaginea. Netezirea inseamna sa se faca media aricmetica a valorilor din filturil 3\*3 si sa se inlocuiasa toate valorile din filtru cu valoarea mediei.   
3. Pasul 3- afiseaza matricea transformata.

Modul de parcurgere a filtrului



Rezultat:

