7. a)

Algoritmul returneaza pozitia lui “v” in sirul ordonat crescator.

7. b)

**[PRECONDITII]**

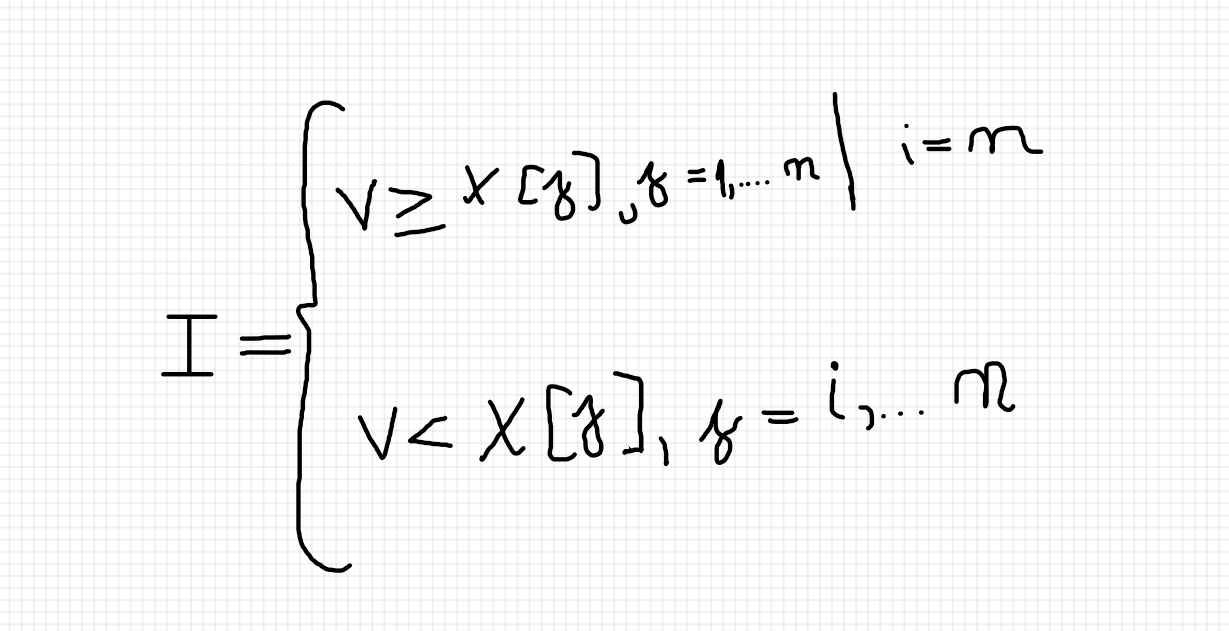
* Tabloul “x” este ordonat crescator cu cel putin 2 elemente
* Variabila “v” care sa respecte tipul elementelor din tablou

**[POSTCONDITII]**

* Pozitia in tabloul ordonat crescator a variabilei “v”

**Verificarea corectitudinii:**

**[INVARIANTUL]**

****

**Argumentare: v x [j], j = 1,…n|i = n**

Aceasta proprietate se refera la situatia in care “i” ramane la valoarea de inceput, adica “n” si variabila “v” este mai mare sau egala decat primul element din tablou. Tabloul fiind ordonat crescator, acest lucru ne spune ca daca “v” este mai mare sau egal decat primul element, atunci sigur va fi mai mare sau egal decat toate elementele din tablou. Aceasta proprietate se intampla daca pe parcurs, variabila “i” are valoarea “n”.

(i)Daca nu se intra in ciclu, atunci aceasta proprietate este adevarat inainte de ciclu, “i” primeste valoarea “n”.

(ii)Pe parcursul ciclului aceasta ramane adevarata(acest lucru nu se va intampla niciodata din moment ce vorbim despre faptul ca valoarea lui “i” nu se schimba, referirea la ceva ce nu se intampla este adevarata).

(iii)Aceasta proprietate ramane valabila si dupa executia ciclului, “i” nu se schimba, deci variabila “v” este mai mare sau egala decat toate elementele din tablou, acest lucru implica postconditia, ca “v” sa fie plasat corect in tabloul ordonat crescator.

**Argumentare: v x [j], j = 1,n**

Aceasta proprietate se refera la faptul cand “v” este strict mai mic decat anumite valori din tablou, iar valoarea lui “i” se schimba.

(i)Invariantul este adevarat inainte de intrarea in ciclu daca se intra in ciclu, nu avem cum sa stim cu ce se va compara “v”, deci intrarea in ciclu, implica faptul ca invariantul este correct

(ii) La fiecare pas, decrementarea lui “i” ne indica ca “v” este strict mai mic de elementul current, alta decrementare a lui “i” ne indica ca “v” este strict mai mic decat urmatorul cel mai mare element si implicit mai mica decat elementul cu o pozitie in spate fata “i”, deci v < x[i] < x[i+1], analog pentru fiecare element, ajungem la v < x[i] < x[i+1] <….x[i+k] < x[n]. Proprietatea ramane adevarata pe tot parcursul ciclului.

(iii) La iesirea din ciclu, valoarea lui “i” ne garanteaza ca “v” este strict mai mic decat “n-i” elemente, oricare ar fi acele elemente intre i si n.

**[FINITUDINEA]**

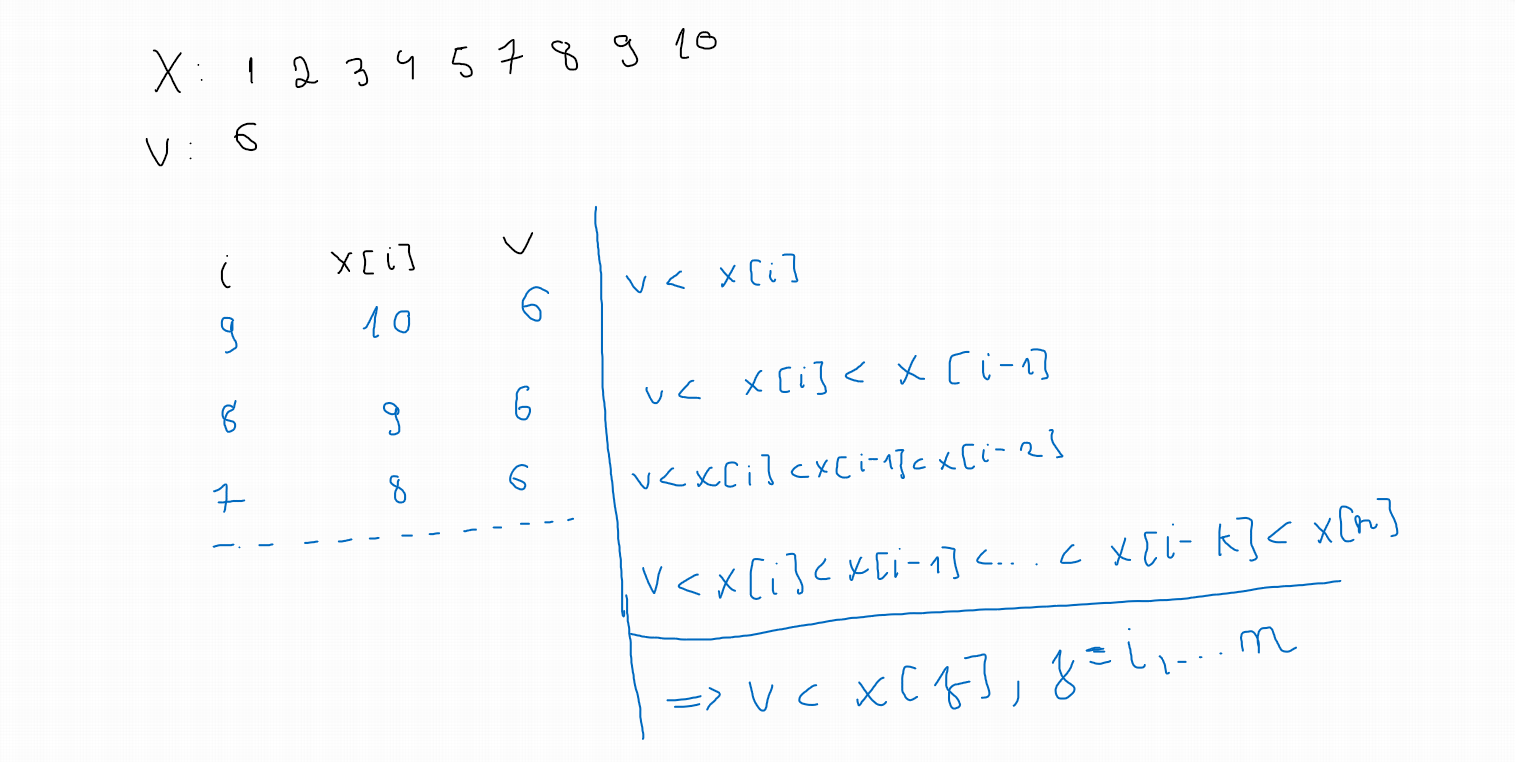
Ciclul porneste de la n si se termina cand “i” este zero, deci face “n+1” pasi.

Numarul de executii pana la terminare este dat de formula: n – (n – i)

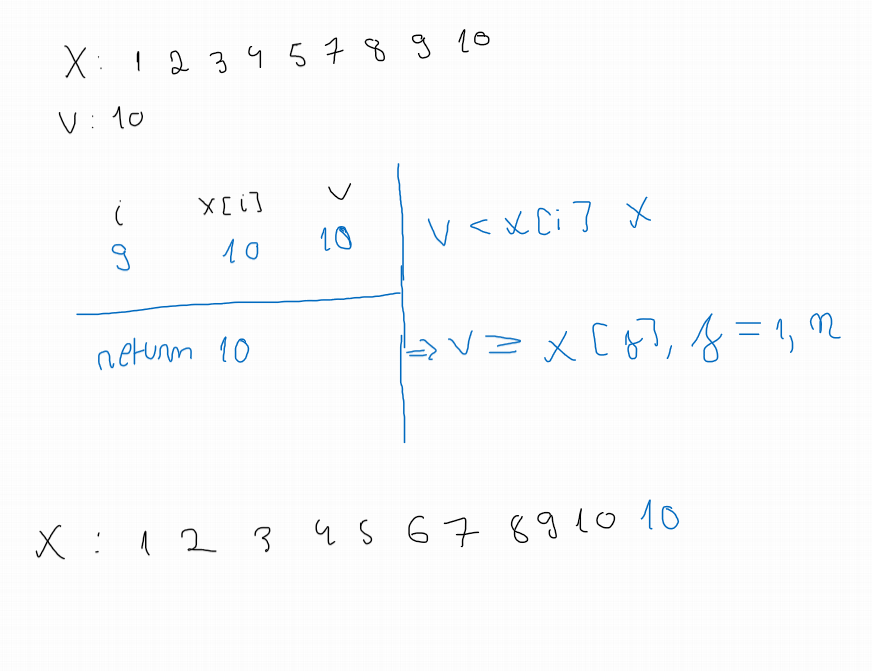
**F:N->N**

**F(t) = n – (n-i)**

**Analiza pentru invariantul 1.**

****

**Analiza pentru invariantul 2.**

****

**Corectitudinea partiala si finitudinea imi asigura corectitudinea algoritmului.**