TAD ListaOrdonata – implementare folosind un arbore binar de cautare.

1. **Enunț**

Studentii unei universitati, indentificati printr-un numar matricol unic, sunt distribuiti in mod aleator la orele de laborator la care trebuie sa participe. Acestia sunt impartiti in mai multe grupe de 5 studenti.

Se cere:

1. Adaugarea unui student.
2. Afisarea tuturor studentilor.
3. Distribuirea pe grupe.
4. Afisarea grupelor.
5. Stergerea unui student.
6. **TAD Multi-Dictionar**

Domeniu :

MD = { md / md este un multi-dictionar cu elemente e = (c, l) , c de tip TCheie, l de tip TLista }

Operatii :

creează(md)

pre**:** true

post**:** mdMD*,* md este dicţionarul vid (fara elemente)

adaugă(md, c, v)

pre:mdMD**,** cTCheie**,** vTLista

post:md’,md’=md+(c, v) (se adaugă în multi-dictionar perechea (c, v))

cauta(md, c, v)

pre:mdMD**,** cTCheie

post:cauta =adevarat dacă (c,v)md, caz în care lTLista e

valoarea asociată cheii c

fals în caz contrar, caz în care l=0*TLista*

şterge(md, c, v)

pre:mdMD**,** cTCheie

post:cTcheie**,** vTElement

perechea (c, v) este ştearsă din multi-dicţionar, dacă cmd

l e noua listă a valorilor asociate cheii c (posibil vidă)

dim(md)

pre:mdD

post: dim=dimensiunea multi-dicţionarului md (numărul de

elemente) N**\***

vid(d)

pre:dD

post: vid= adevărat în cazul în care d e dicţionarul vid

fals în caz contrar

chei(md, m)

pre:mdD

post:mM,m este mulţimea cheilor din multi-dicţionarul md

valori(md, c)

pre:dD

post:cCol*,* c este colecţia valorilor din multi-dicţionarul md

perechi(md, m)

pre:mdMD

post:mM*,* m este mulţimea perechilor (cheie, valoare) din

multi-dicţionarul md

iterator(md, i)

{se creează un iterator pe mult-dicţionarul md}

pre:mdD

post: iI*,* i este iterator pe multi-dicţionarul md

distruge(md)

pre:mdD

post:multi-dicţionarul md a fost 'distrus' (spaţiul de memorie alocat

a fost eliberat)

**TAD Iterator**

Domeniu Iterator:

I={i|i este un iterator pe un Multi-Dictionar avand elemente de tip TElement}

Operatii Iterator:

creeaza(i,md)

element(i,e)

valid(i)

urmator(i)

* creeaza(i,md)

pre: md MD

post: i I, s-a creat iteratorul i pe md (elementul curent din iterator

refera “primul” element din md)

* element(i,e)

pre : i curent este valid (refera un element din md)

post : e TElement, e este elementul curent din iteratie (elementul

din md referit de curent)

* valid(i)

pre:i I

post:- valid = Adevarat, daca curent refera o pozitie valida

= False, altfel

* urmator(i)

pre:i I, curent este valid

post: curent’ refera “urmatorul” element din md fata de cel referit de

curent