Algoritmos y Estructuras de Datos II

Trabajo Práctico 1: Especificación

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Lollapatuza

Y sus puestos de comida

Integrante	LU	Correo electrónico
Gardey, Juan Pablo	1495/21	jpgardey@dc.uba.ar
Fontana Walser, Florencia	1530/21	florfontana02@gmail.com
Rossi, Hernan Guido	791/21	guidorossi1996@gmail.com
Muñoz, Joaquín Eliseo	1484/21	joaquin.e.munoz@gmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

1. Resolución

1.1. Definiciones

TAD PRODUCTO es STRING

TAD CANTIDAD es NAT

TAD PORCENTAJE es NAT $\{porcentaje \leq 100\}$

TAD STOCK es DICC+(PRODUCTO,CANTIDAD)

TAD COMPRA es DICC+(PRODUCTO,CANTIDAD)

TAD MENÚ es DICC+(PRODUCTO,PRECIO)

TAD VENTA es TUPLA(COMPRA, PERSONA, PUESTO)

TAD DESCUENTO es TUPLA(PRODUCTO, CANT, PORCENTAJE)

1.2. Desarrollo

TAD PRECIO

extiende Nat

otras operaciones

div : nat n × nat k ⟶ nat {0 <k}

aplicarDescuento : nat p × nat d ⟶ nat {d <100}

axiomas

div(n,k) ≡ **if** n <k **then** 0 **else** 1+div(n-k,k) **fi**

aplicarDescuento(p, d) ≡ div(p × (100 - d), 100)

Fin TAD

TAD DICCIONARIO+

extiende dicc

géneros dicc+(clave, significado)

otras operaciones

dameClave : dicc(clave, significado) ⟶ clave

axiomas

dameClave(d) ≡ dameUno(claves(d))

Fin TAD

TAD LOLLAPATUZA

usa Nat, Diccioanario+, Conjunto, Puesto, Persona, Producto, Compra

exporta lolla, generadores, observadores, quiénGastóMás

géneros lolla

igualdad observacional

$$(\forall L1, L2 : \text{lolla}) \left(L1 =_{\text{obs}} L2 \iff \begin{pmatrix} \text{puestos(L1)} =_{\text{obs}} \text{puestos(L2)} \wedge \text{personasHabilitadas(L1)} \\ =_{\text{obs}} \text{personasHabilitadas(L2)} \end{pmatrix} \right)$$

observadores básicos

puestos : lolla $\longrightarrow$ conj(puesto)

personasHabilitadas : lolla $\longrightarrow$ conj(persona)

generadores

crearLolla : conj(puesto) c $\times$ conj(persona) $\longrightarrow$ lolla
$\left\{ \begin{matrix} \#(c) > 0 \wedge \#(pers) > 0 \wedge (\forall \text{ p: puesto})(\forall \text{ p': puesto})((p \in c \wedge p' \in c) \rightarrow \text{menuPuesto(p)} = \\ \text{menuPuesto(p'))} \end{matrix} \right\}$

[1]

hackearSistema : lolla l $\times$ producto prod $\times$ persona per $\longrightarrow$ lolla
$\left\{ \begin{matrix} \text{per} \in \text{personasHabilitadas(l)} \wedge (\exists \text{ p: puesto})(p \in \text{puestos(l)} \wedge \text{sePuedeHackearPuesto?(p, prod,} \\ \text{per)} \end{matrix} \right\}$

comprarEnPuesto : lolla l $\times$ puesto p $\times$ persona per $\times$ compra c $\longrightarrow$ lolla
$\{\text{per} \in \text{personasHabilitadas(l)} \wedge p \in \text{puestos(l)} \wedge \text{esCompraValida?(p, c)}\}$

otras operaciones

quiénGastóMás : lolla $\longrightarrow$ persona

quiénGastóMásAux : persona per $\times$ conj(persona) pers $\longrightarrow$ persona
$\{\text{per} \in \text{personasHabilitadas(l)} \wedge \text{pers} \subset \text{personasHabilitadas(l)}\}$

hackearSistemaAux : conj(puesto) $\times$ producto $\times$ persona $\longrightarrow$ conj(puesto)

[1] Los menúes serán iguales, con los mismos precios, pero cambiará el stock según el puesto. Que un puesto no venda un producto significa que su stock inicial será 0.

axiomas

(∀ l: lolla, prod: producto, per: persona, p: puesto, pers: conj(persona), ps: conj(puesto), c: compra)

puestos(crearLolla(ps, pers)) ≡ ps

puestos(hackearSistema(l, prod, per)) ≡ **if** sePuedeHackearPuesto?(dameUno(puestos(l))) **then**
Ag(hackear(p, prod, per), sinUno(puestos(l)))
else
Ag(dameUno(puesto(l)), hackearSistemaAux(sinUno(puestos(l)), prod, per)
fi

puestos(comprarEnPuesto(l, per, p, c) ≡ Ag(comprar(p, c, per), comprarEnPuestoAux(puestos(l), p))

hackearSistemaAux(ps, prod, per) ≡ **if** sePuedeHackear?(dameUno(ps)) **then**
Ag(hackear(dameUno(ps), prod, per), sinUno(ps))
else
Ag(dameUno(ps), hackearSistemaAux(sinUno(ps), prod, per)
fi

personasHabilitadas(crearLolla(ps, pers)) ≡ ps

personasHabilitadas(hackearSistema(l, prod, per)) ≡ personasHabilitadas(l)

personasHabilitadas(comprarEnPuesto(l, per, p, c)) ≡ personasHabilitadas(l)

quienGastóMás(l) ≡ quienGastóMásAux(
l,
dameUno(personasHabilitadas(l)),
sinUno(personasHabilitadas(l))
)

quienGastóMásAux(l, per, pers) ≡ **if** pers = ∅ **then**
per
else
if gastoTotal(per) >gastoTotal(dameUno(pers)) **then**
quienGastóMásAux(l, per, sinUno(pers))
else
quienGastóMásAux(l, dameUno(pers), sinUno(pers))
fi
fi

Fin TAD

TAD PUESTO

igualdad observacional

$$(\forall n, m : \text{puesto}) \left(n =_{\text{obs}} m \iff \begin{pmatrix} \text{menúPuesto(n)} =_{\text{obs}} \text{menúPuesto(m)} \land \text{stockPuesto(n)} \\ =_{\text{obs}} \text{stockPuesto(m)} \land \text{descuentos(n)} =_{\text{obs}} \text{descuentos(m)} \\ \land \text{verVentas(n)} =_{\text{obs}} \text{verVentas(m)} \end{pmatrix} \right)$$

usa Nat, Compra, Venta, Descuento, Menú, Stock, Persona

exporta puesto, generadores, observadores, tieneDescuento?, estáEnMenú?, dameStock, damePrecio, esCompraVálida?, dameDescuento, gastoPersonaEnPuesto, calcularGasto, sePuedeHackear?

géneros puesto

observadores básicos

verVentas : puesto $\longrightarrow$ multiconj(venta)

stockPuesto : puesto $\longrightarrow$ stock

menúPuesto : puesto $\longrightarrow$ menú

descuentos : puesto $\longrightarrow$ conj(descuento)

generadores

nuevoPuesto : menú $\times$ stock $\longrightarrow$ puesto $\{\text{claves(menú)} =_{obs} \text{claves(stock)}\}$

agDescuento : puesto p $\times$ descuento d $\longrightarrow$ puesto
$\{\neg(\exists$ d':descuento)(d' $\in$ descuentos(l) $\land$ π_1(d) = π_1(d') $\land$ π_2(d) = π_2(d'))$\}$

comprar : puesto p $\times$ compra c $\times$ persona $\longrightarrow$ puesto $\{\text{c} \neq \emptyset \land_L \text{esCompraVálida?(p,c)}\ \}$

hackear : puesto $\times$ producto $\times$ persona $\longrightarrow$ puesto

otras operaciones

tieneDescuento? : puesto $\times$ producto $\times$ cantidad $\longrightarrow$ bool

tieneDescuentoAux? : conj(descuento) $\times$ producto $\times$ cantidad $\longrightarrow$ bool

estáEnMenú? : puesto $\longrightarrow$ bool

dameStock : puesto p $\times$ producto prod $\longrightarrow$ nat $\{\text{estáEnMenú?(p, prod)}\}$

damePrecio : puesto p $\times$ producto prod $\longrightarrow$ precio $\{\text{estáEnMenú?(p, prod)}\}$

esCompraValida? : puesto $\times$ compra $\longrightarrow$ bool

dameDescuento : puesto $\times$ producto $\times$ cantidad $\longrightarrow$ porcentaje

dameDescuentoAux : conj(descuento) $\times$ producto $\times$ cantidad $\longrightarrow$ porcentaje

hayMejorDescuento? : conj(descuento) $\times$ descuento $\longrightarrow$ bool

gastoPersonaEnPuesto : puesto $\times$ persona $\longrightarrow$ nat

gastoPersonaEnPuestoAux : multiconj(venta) $\times$ persona $\longrightarrow$ nat

calcularGasto : compra $\times$ menu $\longrightarrow$ nat

sePuedeHackear? : puesto $\times$ producto $\times$ persona $\longrightarrow$ bool

sePuedeHackearAux : puesto $\times$ multiconj(venta) $\times$ producto $\times$ persona $\longrightarrow$ bool

ventaHackeable? : venta $\times$ producto $\times$ persona $\longrightarrow$ bool

hackearAux : multiconj(venta) $\times$ producto $\times$ cantidad $\longrightarrow$ puesto

axiomas (∀ m: menu, s: stock, prod: producto, cant: cantidad, desc: descuento, p: puesto, c: compra, per: persona, d: conj(descuento), v: multiconj(venta))

verVentas(nuevoPuesto(m, s)) ≡ ∅

verVentas(agDescuento(p, desc)) ≡ verVentas(p)

verVentas(comprar(p, c, per)) ≡ Ag(<c, per, p>, verVentas(p))

verVentas(hackear(p, prod, per)) ≡ **if** ventaHackeable?(dameUno(verVentas(p))) **then**
Ag(
<
definir(prod, obtener(prod, π_1(dameUno(verVentas(p)))) - 1),
per,
p
>,
sinUno(verVentas(p))
)
else
Ag(
dameUno(verVentas(p)),
hackearAux(sinUno(verVentas(p))
)
fi

menúPuesto(nuevoPuesto(m, s)) ≡ m

menúPuesto(agDescuento(p, desc)) ≡ menúPuesto(p)

menúPuesto(comprar(p, c, per)) ≡ menúPuesto(p)

menúPuesto(hackear(p, prod, per)) ≡ menúPuesto(p)

stockPuesto(nuevoPuesto(m, s)) ≡ s

stockPuesto(agDescuento(p, desc)) ≡ stockPuesto(p)

stockPuesto(comprar(p, c, per)) ≡ **if** ∅?(c) **then**
stockPuesto(p)
else
def(
dameClave(c),
dameStock(p, dameClave(c)) - obtener(dameClave(c), c),
stockPuesto(comprar(p, borrar(dameClave(c), c), per)
)
fi

stockPuesto(hackear(p, prod, per)) ≡ definir(prod, dameStock(p, prod) + 1, stockPuesto(p))

hackearAux(v, prod, cant) ≡ **if** ventaHackeable?(dameUno(v)) **then**
Ag(
<
definir(
prod,
obtener(prod, π_1(dameUno(v))) - 1
),
per,
p
>,
sinUno(v)
)
else
Ag(dameUno(v), hackearAux(sinUno(v), prod, cant)
fi

descuentos(nuevoPuesto(m, s)) ≡ ∅

descuentos(agDescuento(p,desc))) ≡ Ag(desc, listarDescuentos(p))

descuentos(comprar(p, c, per)) ≡ descuentos(p)

descuentos(hackear(p, prod, per)) ≡ descuentos(p)

```
tieneDescuento?(p, prod, cant) ≡ if descuentos(p) = ∅ then
                                     false
                                 else
                                     if
                                         prod = π1(dameUno(descuentos(p))) ∧
                                         cant ≥ π2(dameUno(descuentos(p)))
                                     then
                                         true
                                     else
                                         tieneDescuentoAux?(sinUno(descuentos(p)), prod, cant)
                                     fi
                                 fi
tieneDescuentoAux?(d, prod, cant) ≡ if d = ∅ then
                                        false
                                    else
                                        if
                                            prod = π1(dameUno(d)) ∧
                                            cant ≥ π2(dameUno(d))
                                        then
                                            true
                                        else
                                            tieneDescuentoAux?(sinUno(d), prod, cant)
                                        fi
                                    fi
```

estáEnMenú?(p, prod) ≡ def?(prod, menuPuesto(p)

dameStock(p, prod) ≡ obtener(prod, stockPuesto(p))

damePrecio(p, prod) ≡ obtener(prod,menúPuesto(p))

```
esCompraVálida?(p, c) ≡ if c = ∅ then
                            true
                        else
                            if
                                ¬estáEnMenú?(p, dameClave(c)) ∨
                                obtener(dameClave(c), c) > dameStock(p, dameClave(c))
                            then
                                false
                            else
                                esCompraVálida?(p, borrar(dameClave(c), c))
                            fi
                        fi
dameDescuento(p, prod, cant) ≡ if descuentos(p) = ∅ then
                                   0
                               else
                                   if  π1(dameUno(descuentos(p)) = prod ∧
                                   ¬hayMejorDescuento? (descuentos(p), dameUno(descuentos(p)))
                                   then
                                       π3(dameUno(descuentos(p))
                                   else
                                       dameDescuentoAux(sinUno(descuentos(p), prod, cant)
                                   fi
                               fi
```

```
dameDescuentoAux(d, prod, cant) ≡ if vacío?(d) then
                                      0
                                  else
                                      if
                                          π1(dameUno(d) = prod ∧
                                          ¬hayMejorDescuento?(d, dameUno(d)
                                      then
                                          π3(dameUno(d))
                                      else
                                          dameDescuentoAux(sinUno(d), prod, cant)
                                      fi
                                  fi
hayMejorDescuento?(d, desc) ≡ if vacio?(d) then
                                  false
                              else
                                  if π1(dameUno(d)) = π1(desc) ∧ π2(dameUno(d)) >π2(desc) then
                                      true
                                  else
                                      hayMejorDescuento?(sinUno(d), desc)
                                  fi
                              fi
gastoPersonaEnPuesto(p, per) ≡ gastoPersonaEnPuestoAux(verVentas(p), per)
gastoPersonaEnPuestoAux(v, per) ≡ if vacío?(v) then
                                      0
                                  else
                                      if π2(dameUno(v)) = per then
                                          calcularGasto(π1(dameUno(v)))
                                          + gastoPersonaEnPuestoAux(sinUno(v), per)
                                      else
                                          gastoPersonaEnPuestoAux(sinUno(v), per)
                                      fi
                                  fi
calcularGasto(c) ≡ if c = ∅ then
                       0
                   else
                       obtener(dameClave(c),c) x aplicarDescuento(
                           damePrecio(dameClave(c)),
                           dameDescuento(dameClave(c),
                           obtener(dameClave(c),c)
                       )
                       + calcularGasto(borrar(dameClave(c), c))
                   fi
dameStock(hackear(p, prod, per), prod) ≡ dameStock(p, prod) + 1
sePuedeHackearPuesto(p)? ≡ sePuedeHackearAux(verVentas(p)
```

```
sePuedeHackearAux(p, v, prod, per) ≡ if vacío?(v) then
                                        false
                                     else
                                        if
                                            def?(prod, dameUno(v)) ∧
                                            ¬tieneDescuento?(
                                                p,
                                                prod,
                                                obtener(prod ,π1(dameUno(v)))
                                            )
                                        then
                                            true
                                        else
                                            sePuedeHackearAux?(p, sinUno(v), prod, per)
                                        fi
                                     fi
ventaHackeable?(p, v, prod, cant, per) ≡ ¬tieneDescuento?(descuentos(p), prod, cant) ∧
                                         def?(prod, π1(v)) ∧
                                         π3(v) = per
```

Fin TAD

TAD PERSONA

usa Nat, Conjunto, Compra, Puesto

exporta persona, observadores, generadores, gastoTotal

géneros persona

igualdad observacional

$(\forall p1, p2 : \text{persona})\ (p1 =_{\text{obs}} p2 \iff (\text{idPersona(p1)} =_{\text{obs}} \text{idPersona(p2)}))$

generadores

nuevaPersona : nat $\longrightarrow$ persona

observadores básicos

idPersona : persona $\longrightarrow$ nat

otras operaciones

gastoTotal : persona $\times$ puesto $\longrightarrow$ conj(compra)

axiomas ($\forall$ n: nat, per: persona, pues: conj(puesto)

idPersona(nuevaPersona(n)) $\equiv$ n

gastoTotal(per, pues) $\equiv$ **if** pues $= \emptyset$ **then**
0
else
gastoPersona(dameUno(pues), per) + gastoTotal(per, sinUno(pues))
fi

Fin TAD