Lab 3

Date Submitted: 03-29-2023

Name: Dylan Brown

COSC 3319 Section 01

Meeting Days: MWF

Grading Option: A

IO Redirection used:

linkedlisttransactions<test.txt>results

Results for my Lab

List size: 5

list Contents:

Car made by: Chev Number of Doors: 3

Car made by: Ford Number of Doors: 2

Car made by: Ford Number of Doors: 4

Car made by: GMC Number of Doors: 2

Car made by: RAM Number of Doors: 2

List size: 4

List Content:

Car made by: Chev Number of Doors: 3

Car made by: Ford Number of Doors: 4

Car made by: GMC Number of Doors: 2

Car made by: RAM Number of Doors: 2

List Contents:

Plane made by: Cessna Number of doors: 4 Number of Engines: 4

Plane made by: Piper Number of doors: 2 Number of Engines: 1

Plane made by: Boeing Number of doors: 3 Number of Engines: 6

Car made by: Chev Number of Doors: 3

Car made by: Ford Number of Doors: 4

Car made by: GMC Number of Doors: 2

Car made by: RAM Number of Doors: 2

LinkedList.ads

with Ada.Unchecked\_Deallocation;

generic

type element is tagged limited private;

with function "="(I,E: element'Class) return Boolean; --overloading = to compare equality between multiple objects

package Linkedlist is

type DList is private;

type elementPtr is access all element'Class;

procedure makeHead(list: in out DList);

function ListSize(list: in DList) return Integer;

procedure InsertAfter(list: in out DList; Item: in elementPtr);

procedure InsertBefore(list: in out DList; Item: in elementPtr);

generic

with procedure Process(I : element'Class);

procedure ManuealMove(list : in DList);

function FindItem(list : in DList; C : elementPtr) return elementPtr;

function Delete(list : in out DList; C : elementPtr) return elementPtr;

private

type Node;

type nodePtr is access Node;

type Node is record

Data : elementPtr;

Next, prev : nodePtr;

end record;

type DList is record

count : Integer := 0;

Head : nodePtr;

end record;

end Linkedlist;

LinkedList.adb

with Ada.Text\_IO; use Ada.Text\_IO;

with Ada.Unchecked\_Deallocation;

package body Linkedlist is

procedure Free is new Ada.Unchecked\_Deallocation(node, nodePtr);

procedure makeHead(list : in out DList) is -- makes an empty list that points back at intsself

Head : nodePtr := new node;

begin

list.count := 0;

Head.Next := Head;

list.Head := Head;

head.data := null;

head.prev := head;

end;

function ListSize(list: in DList) return Integer is

begin

return list.count;

end ListSize; -- get number of items in list

procedure InsertAfter(list: in out DList; Item: in elementPtr) is

newNode : nodePtr := new node;

begin

newNode.Data := Item;

newNode.next := list.head.next;

newNode.prev := list.head;

newNode.next.prev := newNode;

newNode.prev.next := newNode;

list.count := list.count + 1;

end;

procedure InsertBefore(list: in out DList; Item: in elementPtr) is

newNode : nodePtr := new node;

begin

newNode.Data := Item;

newNode.prev := list.head.prev;

newNode.next := list.head;

newNode.prev.next := newNode;

newNode.next.prev := newNode;

list.count := list.count + 1;

end;

procedure ManuealMove(list : in DList) is

newNode: nodePtr;

begin

NewNode := list.Head.Next;

while newNode /= list.Head loop

Process(newNode.Data.all);

newNode := newNode.Next; New\_Line;

end loop;

end;

function FindItem(list : in DList; C : elementPtr) return elementPtr is

newNode: nodePtr;

begin

newNode := list.Head.Next;

while newNode /= list.Head loop

if C.all = newNode.Data.all then

return newNode.Data;

end if;

newNode := newNode.Next;

end loop;

return null;

end;

function Delete(list : in out DList; C : elementPtr) return elementPtr is

newNode: nodePtr; pt : elementPtr;

begin

newNode := list.Head.Next;

while newNode /= list.Head loop

if C.all = newNode.Data.all then

newNode.prev.Next := newNode.Next;

newNode.Next.prev := newNode.prev;

pt := newNode.Data; Free(newNode);

list.count := list.count - 1;

return pt;

end if;

newNode := newNode.Next;

end loop;

return null;

end;

end Linkedlist;

MakeCar.ads

MakeCar.adb

MakePlane.ads

MakePlane.adb

linkedListTransactions.adb (Driver)

with ada.Text\_IO; use ada.Text\_IO;

with LinkedList;

procedure LinkedListtransactions is

package intlist is new LinkedList(Integer);

use intlist;

I:Integer;

begin

I:=1;

makeHead(1);

insertAfter(33,I);

insertAfter(57,I);

insertbefore(85,I);

insertAfter(95,I);

for k in Integer range 1 .. I-1 loop

put(Integer'Image(print(K)));

end loop; New\_Line(1);

Delete(57,I);

Delete(33,I);

delete(33,I);

insertAfter(22,I);

Delete(95,I);

for k in Integer range 1 .. I-1 loop

put(Integer'Image(print(K)));

end loop;

end Linkedlisttransactions;