אביב 2011 מועד א **פתרון**

1. ADT

# פתרון

**חלק 1:**

typedef void\* ElementP;

typedef struct ReducibleList\* ReducibleListP;

typedef ElementP (\*ReduceElementsFP)(ElementP, ElementP);

typedef void (\*DeleteElementFP)(ElementP);

void addElement(ReducibleListP list, ElementP elem);

ElementP getElement(ReducibleListP list, int index);

void destroyList(ReducibleListP list);

// createReducibleList was already given to the students

ReducibleListP createReducibleList(int maxElements, ReduceElementsFP reduceFunc, DeleteElementFP deleteFunc);

**חלק 2:**

void addElement(ReducibleListP list, ElementP elem){

ElementP newElement;

int i;

if (list->numElements == list->maxElements){

newElement=list->reduceFunc(

list->elements[list->maxElements-1]

,list->elements[list->maxElements-2]);

list->deleteFunc(list->elements[list->maxElements-1]);

list->deleteFunc(list->elements[list->maxElements-2]);

list->elements[list->maxElements-1]=newElement;

} else {

list->numElements++;

if (list->numElements > 1)

list->elements[list->numElements-1] =

list->elements[list->numElements-2];

}

for (i=list->numElements-2;i>0;i--)

list->elements[i]=list->elements[i-1];

list->elements[0]=elem;

}

**חלק 3:**

ReducibleListP balanceList = createReducibleList(13,reduceBalance,deleteBalance);

ElementP reduceBalance(ElementP el1, ElementP el2){

Balance\* bl1 = (Balance\*) el1;

Balance\* bl2 = (Balance\*) el2;

Balance\* result = createBalance(bl1->income+bl2->income,

bl1->expense+bl2->expense);

return result;

}

void deleteBalance(ElementP el){

free(el);

}

1. **C++**

**פתרון:**



#ifndef \_SOLUTION\_H\_

#define \_SOLUTION\_H\_

#include "list.h"

class Animal : public CListItem

{

public:

Animal(const char\* name, int id);

virtual ~Animal();

int GetKey() { return \_id; }

virtual void Print(ostream& os) const = 0;

protected:

char\* \_name;

int \_id;

};

class Zoo

{

public:

static unsigned int GetNumOfAnimals();

Zoo& operator+=(Animal\* pa) {

\_pList.Insert(pa);

return \*this; /\*it's ok also to return void\*/

};

friend ostream& operator<<(ostream& os, const Zoo& pZoo);

Animal\* operator()(int id);

protected:

CList \_pList;

static unsigned int \_total;

};

ostream& operator<<(ostream& os, const Animal& pa)

{

pa.Print(os);

return os;

}

class Mammal : public Animal

{

public:

Mammal(const char\* name, int id);

void Print(ostream& os) const;

};

#endif

1. **C++ מה יודפס**

1

A::A()

2

A::A(A&)

3

B::B()

4

A::A()

B::B()

5

A::operator=(cont A&)

6

A::A(A&)

A::A(A&)

B::B(T)

~A::A()

7

B::B(T)

A::A()

B::B()

C::C(int)

8

B::B(T)

B::B()

C::C(int)

9

10

~A::A()

~A::A()

~A::A()

~A::A()

~A::A()

2. BASH

א.

grep faculty \* | grep Technion | sort |uniq

ב.

cat users | cut – f 5 –d":" | cut –f1 –d" "

ג.

add\_dates

cat file | add

add

#! /bin/bash

while read line; do

echo -n $line

if [[ $line == \*date\* ]]; then

echo `date`

fi

done

שאלות הבנה

1. מערכת הפעלה יוניקס, שפת תיכנות JAVA, פיתוח במודל איטרטיבי.
2. לסובייקט יש רשימת משקיפים שכל משקיף המעוניין להתעדכן נרשם אליה. כאשר משתנה מצבו של הסובייקט הוא עובר על הרשימה הנ"ל ומפעיל את המתודה update של כל האובייקטים ברשימה.
3. אם הפונקציה Shout זורקת חריגה, לא יתבצע שחרור הזכרון.

הפתרון: להשתמש בפוינטר חכם, המשחרר בעצמו את הזכרון כשיוצאים מהבלוק או כשנזרקת חריגה.

1. המשפט delete q ימחוק את הזיכרון של p והוא יהפוך לפוינטר מתנדנד.

מניעת התופעה: שימוש בפוינטר חכם אשר ביצע חפיפה של "operator ="