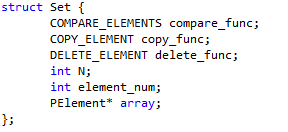
אביב 2013 מועד א - פתרון

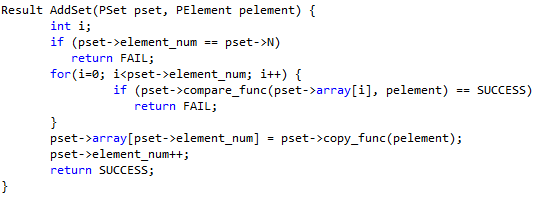
שאלה 1: ADT

פתרון:

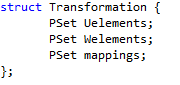
1. **מימוש המבנה:**

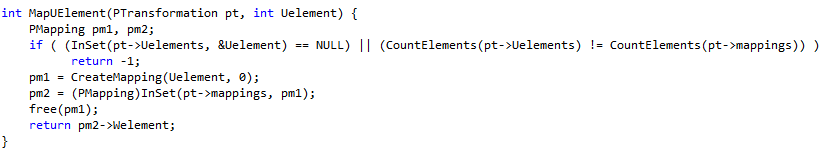
****

1. **מימוש הפונקציה:**

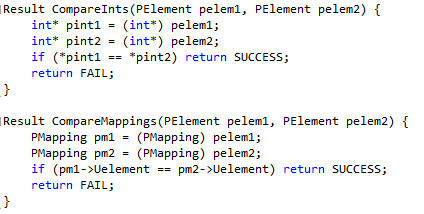
****

1. מימוש המבנה:

  
ד. מימוש הפונקציה:



מימוש פונקציות המשתמש:



שאלה 2: C++

סעיף א'

class Account {

public:

// C'tors

Account(const char\* name, const char\* surname, unsigned int id, double balance=0);

// D'tor

virtual ~Account();

// Const methods - get members

const char\* getName() const {return \_name;};

const char\* getSurname() const {return \_surname;};

unsigned long getID() const {return \_id;};

double getBalance() const {return \_balance;};

virtual double getInterestRate() const = 0;

// Operators

virtual Account& operator+=(double amount);

virtual Account& operator-=(double amount);

virtual void print (ostream& rhs) const = 0;

friend ostream& operator<<(ostream& lhs, const Account& rhs);

protected:

char\* \_name, \*\_surname;

unsigned long \_id;

double \_balance;

};

סעיף ב'

class RegularAccount : public Account {

public:

// C'tors

RegularAccount(const char\* name, const char\* surname, unsigned int id,

double maxOverdraft, double balance=0);

// Const methods - get members

double getInterestRate() const {return \_interestRate;};

static double setInterestRate(double rate) {\_interestRate = rate;};

// Operators

Account& operator-=(double amount) throw (OVERDRAFT\_ERROR);

void print (ostream& rhs) const;

private:

static double \_interestRate;

double \_maximumOverdraft;

};

class PremiumAccount : public Account {

public:

// C'tors

PremiumAccount(const char\* title, const char\* name, const char\* surname,

unsigned int id, double interestRate, double balance=0);

// Const methods - get members

double getInterestRate() const {return \_interestRate;};

double setInterestRate(double rate) {\_interestRate = rate;};

// Operators

Account& operator+=(double amount);

void print (ostream& rhs) const;

// D'tor

~PremiumAccount() {delete[] \_title;};

private:

char\* \_title;

double \_interestRate;

};

סעיף ג'

ostream& operator<<(ostream& lhs, const Account& rhs) {

rhs.print(lhs);

return lhs;

}

המתודות שעשויות להיקרא (ע"י מנגנון ההחלפה):

void RegularAccount::print (ostream& rhs) const;

void PremiumAccount::print (ostream& rhs) const;

סעיף ד'

void embezzle(Account\* accounts[], unsigned int numAccounts,

PremiumAccount& destAccount) {

double quantity;

for (unsigned int i=0; i<numAccounts; i++) {

quantity = accounts[i]->getBalance() -

(int) accounts[i]->getBalance();

quantity = (quantity<0?1+quantity:quantity);

try {

(\*accounts[i])-=quantity;

destAccount+=quantity;

}

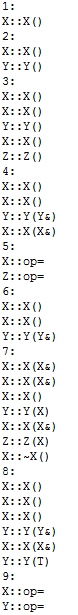
catch (OVERDRAFT\_ERROR e) { };

}

}

שאלה 3: מה יודפס?

**פתרון:**

****

****

שאלה 4: BASH והבנה

1.

#!/bin/bash

word=$1

dir=$2

num=$3

for file in $dir/\*; do

if [[ -f $file ]]; then

count=`cat $file|grep "$word"|wc -l`

if [[ $count -gt $num ]]; then

echo $file $word $count

fi

fi

done

2.

doube\_it

#!/bin/bash

cat $1 | ./double > $2

double

#!/bin/bash

while read line; do

echo $line

echo $line

done

שאלות הבנה: פתרון

1. כדי שלכולן יהיה ממשק משותף – המתודה update, והסובייקט יוכל להכיר רק ממשק אחד ולעבוד בעזרתו עם כל האובזרוורים.
2. מאחר והקונסטרקטור הוצהר private למניעת ייצור אובייקטים, יש לאפשר למחלקה עצמה לייצר את האובייקטים והדרך היחידה לקרוא למתודה מהמחלקה היא דרך מתודה סטטית.
3. Aggregation : הכלה לא חזקה, כל אובייקט מוכל יכול להתקיים בפני עצמו.

Composition: הכלה חזקה, האובייקטים המוכלים אינם יכולים להתקיים בפני עצמם.

1. יש לשים את המימוש בקובץ .h . הסיבה: מאחר והמנגנון הוא מנגנון מקרו (העתקת קוד) ומאחר והקומפיילר מקמפל כל קובץ C בנפרד, הקומפיילר דורש לראות את מימוש המתודות באותו קובץ C שבו יש קריאה למתודה. אם נשים את המימוש בקבצי C, נקבל שכפול קוד. לכן, נשים בקובץ H ונעשה include שלו בכל קובץ C שצריך את המימוש.