חורף 2017 מועד א פתרון

**פתרון שאלה 1:** פתרון סעיף 1:

/\*User functions\*/

**typedef** PElem (\*KEY\_CLONE\_FUNC)(PElem);

**typedef** **void** (\*KEY\_DEL\_FUNC)(PElem);

**typedef** Bool (\*KEY\_CMP\_FUNC)(PElem, PElem);

**typedef** **void** (\*VALUE\_DEL\_FUNC)(PElem);

**typedef** PElem (\*VALUE\_CLONE\_FUNC)(PElem);

**struct** Pair\_{

PElem pKey;

PElem pValue;

KEY\_CLONE\_FUNC keyCloneFunc;

KEY\_DEL\_FUNC keyDelFunc;

KEY\_CMP\_FUNC keyCmpFunc;

VALUE\_DEL\_FUNC valueDelFunc;

VALUE\_CLONE\_FUNC valueCloneFunc;

};

פתרון סעיף 2:

PElem **DoubleClone**(PElem pDouble)

{

**double**\* pNewDouble = (**double**\*)**malloc**(**sizeof**(**double**));

\*pNewDouble = \*((**double**\*)pDouble);

**return** pNewDouble;

}

PElem **StringClone**(PElem pString)

{

**char**\* newString = (**char**\*)**malloc**(**strlen**((**char**\*)pString)+1);

**strcpy**(newString, (**char**\*)pString);

**return** newString;

}

פתרון סעיף 3:

**void** **PairDestroy**(PPair pPair)

{

**if**(pPair == NULL)

{

**return**;

}

pPair->keyDelFunc(pPair->pKey);

pPair->valueDelFunc(pPair->pValue);

**free**(pPair);

}

PElem **PairGetValue**(PPair pPair)

{

**if**(pPair == NULL)

{

**return** NULL;

}

**return** pPair->valueCloneFunc(pPair->pValue);

}

פתרון סעיף 4:

**void** **listDestory**(PList pList)

{

**if**(pList == NULL)

{

**return**;

}

PNode pNode = pList->head;

**while**(pNode != NULL)

{

PNode tmp = pNode->next;

pList->delFunc(pNode->pElem);

**free**(pNode);

pNode = tmp;

}

**free**(pList);

}

פתרון סעיף 5:

PElem **DictionaryPairCopy**(PElem pElem)

{

**return** PairCopy(pElem);

}

**void** **DictionaryPairDestroy**(PElem pElem)

{

PairDestroy(pElem);

}

Bool **DictionaryPairCompare**(PElem pElem1, PElem pElem2)

{

**return** PairCompare(pElem1,pElem2);

}

pDictionary->pPairList = ListCreate(DictionaryPairCopy, DictionaryPairDestroy, DictionaryPairCompare);

# פתרון שאלה 2

#ifndef DPT\_STORE

#define DPT\_STORE

#include<iostream>

#include<exception>

#include<string>

#include<map>

#include<set>

class ErrorMessage : public exception

{

public:

ErrorMessage(string msg) : msg\_(msg) {}

const char\* what() const throw()

{

return msg\_.c\_str();

}

private:

string msg\_;

};

class DptStore

{

public:

DptStore(const string& storeName) : storeName\_(storeName)

{

if (storeNamesSet.find(storeName) != storeNamesSet.end())

throw ErrorMessage("Store name already exists");

storeNamesSet.insert(storeName);

}

~DptStore()

{

auto nameIterator = storeNamesSet.find(storeName\_);

storeNamesSet.erase(nameIterator);

}

void AddItem(const string& prodNum, int price)

{

bool itemExists;

try

{

int& prodPrice = storeItemsMap.at(prodNum);

itemExists = true;

}

catch (...)

{

itemExists = false;

}

if (itemExists == false)

{

if (price <= 0)

throw ErrorMessage("Please specify a price greater than 0 when creating a new item");

else

storeItemsMap[prodNum] = price;

}

try

{

inventory.at(prodNum)++;

}

catch (...)

{

inventory[prodNum] = 1;

}

}

void SellItem(const string& prodNum)

{

try

{

int& itemInventory = inventory.at(prodNum);

int price = storeItemsMap.at(prodNum);

if (itemInventory == 0)

throw "No Item";

--itemInventory;

cout << "Sold ProdNum: " << prodNum << " for " << price << " dollars" << endl;

}

catch (...)

{

throw ErrorMessage("Item not in stock or doesn't exist");

}

}

private:

static set<string> storeNamesSet;

static map<string, int> storeItemsMap;

map<string, int> inventory;

string storeName\_;

};

set<string> DptStore::storeNamesSet;

map<string, int> DptStore::storeItemsMap;

#endif // !DPT\_STORE

**פתרון שאלה 3**

חלק א: פתרון:

פתרון:

1)

class Plant {

public:

Plant (int,double);

virtual void print(ostream&) const;

virtual void operator++(int)=0;

protected:

int age\_;

double height\_;

};

2)

void Plant::print(ostream& os) const {

os << "height: " << height\_ << endl << "age: " << age\_ << endl;

}

ostream& operator<<(ostream& os, const Plant& p) {

p.print(os);

return os;

}

class Tree : public Plant {

public:

Tree(int=1,double=1);

virtual void operator++(int);

virtual void print(ostream&) const;

};

void Tree::print(ostream& os) const{

os << "Printing Tree" << endl;

Plant::print(os);

}

class Flower : public Plant {

public:

Flower(int=0,double=0);

virtual void operator++(int);

virtual void print(ostream&) const;

};

void Flower::print(ostream& os) const{

os << "Printing Flower" << endl;

Plant::print(os);

}

חלק ב' : פתרון:

1) חסר: בנאי העתקה, דיסטרקטור, ואופרטור השמה. אי מימוש מתודות אלו יגרור שימוש במימוש הדיפולטי שלהם אשר יגרום לדליפות זיכרון ולגישות לא חוקיות לזיכרון.

2)

Smartphone::Smartphone(const Smartphone& rhs): Company\_(createcopy(rhs.Company\_)), Frequency\_(rhs.Frequency\_) {}

Smartphone::~Smartphone(){

delete Company\_;

}

Smartphone& Smartphone::operator=(const Smartphone& rhs) {

if (this != &rhs) {

delete Company\_;

Company\_ = createcopy(rhs.Company\_);

RAM\_ = rhs.RAM\_;

Memory\_ = rhs.Memory\_;

Frequency\_ = rhs.Frequency\_;

}

return \*this;

}

**שאלה 4 – פתרון**

**חלק ראשון: BASH (נק')**

1. שורת ה-BASH:

grep -v id grades | sort -n -r | sort -n -r -k7 | head -3 |

cut -d" " -f1

1. הסקריפט:

#! /bin/bash

if (($# != 1)); then

exit 1

elif [[ -d "$1" ]]; then

rm -f ids

for file in "$1"/\*; do

if [[ -f "$file" ]]; then

grep -v id "$file" | cut -d" " -f1 >> ids

fi

done

if [[ -f ids ]]; then

sort ids | uniq

fi

rm -f ids

else

exit 1

fi

**חלק שני: שאלות הבנה (נק')**

הפקודות המתבצעות:

rm -f prog test \*.o

cc -Wall -c -o main.o main.c

cc -Wall -c -o f1.o f1.c

cc -Wall -c -o f2.o f2.c

cc -Wall -o prog main.o f1.o f2.o