**אביב 2014 מועד ב'**

1. **פתרון שאלת ADT –**

**סעיף א'**

typedef PElement (\*CloneElementFunction)(PElement);

typedef void (\*DelElementFunction)(PElement);

typedef PKey (\*CloneKeyFunction)(PKey);

typedef void (\*DelKeyFunction)(PKey);

typedef PElementKey (\*GetElementKeyFunction)(PElement);

typedef Bool (\*CompareElementKeysFunction)(const PElementKey, const PElementKey);

typedef Bool (\*CompareKeysFunction)(const PKey, const PKey);

typedef void (\*PrintElementFunction)(PElement);

typedef void (\*PrintKeyFunction)(PKey);

typedef struct \_ml

{

PListNode lists;

CloneElementFunction cloneElementFunc;

DelElementFunction deleteElementFunc;

CloneKeyFunction cloneKeyFunc;

DelKeyFunction deleteKeyFunc;

GetElementKeyFunction getElementKeyFunc;

CompareElementKeysFunction cmpElementKeyFunc;

CompareKeysFunction cmpKeyFunc;

PrintElementFunction printElementFunc;

PrintKeyFunction printKeyFunc;

} ML;

**סעיף ב'**

PML MLCreate(CloneElementFunction,

DelElementFunction,

CloneKeyFunction,

DelKeyFunction,

GetElementKeyFunction,

CompareElementKeysFunction,

CompareKeysFunction,

PrintElementFunction,

PrintKeyFunction);

**סעיף ג'**

PElement MLFind(PML pml, PKey key, PElementKey elementKey)

{

if(pml == NULL || key == NULL || elementKey == NULL) return FAILURE;

for( PListNode node = pml->lists; node != NULL; node = node->next )

{

if( pml->cmpKeyFunc( key, node->key ) )

{

for( PElementNode eNode = node->list; eNode != NULL; eNode = eNode->next; )

{

if( pml->cmpElementKeyFunc( elementKey, pml->getElementKeyFunc(eNode->pe) ) )

{

return eNode->pe;

}

}

return NULL;;

}

}

return NULL;

}

**סעיף ד'**

Bool compareClasses(const PKey class1, const PKey class2)

{

if( NULL == class1 || NULL == class2 ) return FALSE;

Class\* pClass1 = (Class\*) class1;

Class\* pClass2 = (Class\*) class2;

if( ( pClass1->className == pClass2->className ) &&

( pClass1->classNumber == pClass2->classNumber ) ) return TRUE;

return FALSE;

}

PElementKey getStudentID(PElement stud)

{

if( NULL == stud ) return NULL;

Student\* pStudent = (Student\*)stud;

return &(pStudent->id);

}

Bool compareIDs(const PElementKey key1, const PElementKey key2)

{

if( NULL == key1 || NULL == key2 ) return FALSE;

int k1 = \*((int\*)key1);

int k2 = \*((int\*)key2);

if( k1 == k2 ) return TRUE;

return FALSE;

}

**פתרון שאלה 2**

**סעיף א'**

template<class Key, class Val>

Val& Dict<Key, Val>::operator[](const Key& NodeKey) {

Node<Key, Val>\* pTmp = pHead\_;

while(pTmp != NULL)

{

if(pTmp->GetKey() == NodeKey)

return pTmp->GetVal();

pTmp = pTmp->GetNext();

}

pTmp = new Node<Key, Val>(NodeKey, new Val(), pHead\_);

pHead\_ = pTmp;

return pHead\_->GetVal();

}

**סעיף ב'**

template<class Key, class Val>

void Dict<Key, Val>::RemoveNode(const Key& NodeKey){

Node<Key, Val>\* pCurr = pHead\_;

if (pHead\_ != NULL)

{

if(pHead\_->GetKey() == NodeKey)

{

pHead\_ = pHead\_->GetNext();

delete pCurr;

return;

}

}

while(pCurr->GetNext() != NULL)

{

Node<Key, Val>\* pNext = pCurr->GetNext();

if(pNext->GetKey() == NodeKey)

{

pCurr->SetNext(pNext->GetNext());

delete pNext;

return;

}

else

pCurr = pCurr->GetNext();

}

}

**סעיף ג'**

class Underflow{};

template<class Val>

class Stack: public Dict<int, Val> {

public:

Stack(): Dict(), size\_(0) {};

void Push(const Val& value) {

(\*this)[size\_] = value;

size\_++;

}

Val Pop() {

if(size\_ > 0)

{

Val PopVal = (\*this)[--size\_];

RemoveNode(size\_);

return PopVal;

}

else

throw Underflow();

}

private:

int size\_;

};

**3. פתרון שאלת מה יודפס:**

**Stage 1:**

A::A()

B::B(int)

A::A()

B::B()

C::C()

**Stage 2:**

B::op++()

A::A(A&)

B::B(B&)

A::A()

B::B(int)

C::C(int)

**Stage 3:**

B::f(B&)

A::A(A&)

B::B(B&)

B::~B()

B::~A()

B::f(B&)

A::A(A&)

B::B(B&)

B::~B()

B::~A()

**Stage 4:**

B::h(A&)

A::A(A&)

A::~A()

C::h(A&)

A::A(A&)

A::~A()

**Stage 5:**

B::op+(B&,B&)

A::A(A&)

B::B(B&)

A::A(A&)

B::~B()

A::~A()

**Stage 6:**

A::A()

B::B(int)

C::C(int)

C::op+(C&)

A::A()

B::B()

C::C(C&)

C::~C()

B::~B()

A::~A()

**Stage 7:**

B::~B()

A::~A()

C::~C()

B::~B()

A::~A()

**Stage 8:**

C::~C()

B::~B()

A::~A()

A::~A()

C::~C()

B::~B()

A::~A()

B::~B()

A::~A()

שאלה 4: פתרון

#!/bin/bash

for f in \*[.cC]; do

if [[ `wc -l $f` > 6 ]]; then

cat $f | head -5 | tail -1

fi

done

ב.

#!/bin/bash

while read line; do

if [[ ${line} == "#include <string.h>" ]]; then

echo " #include <String> "

echo "using namespace std;"

else

echo $line

fi

done

פתרון שאלות ההבנה

1. המצביע החכם הינו אובייקט שנהרס כאשר הוא יוצא מהתחום והזכרון משתחרר ע"י הדיסטרקטור שלו.
2. כדי שלא יהיה ניתן ליצור אובייקטים בדרך הרגילה, אלא רק דרך המתודה הסטטית שלו השולטת במספר האובייקטים שנוצרו (אובייקט בודד).
3. נחסך זמן ריצה הנובע מקריאה לפונקציה, במקום זה מנגנון המקרו מבצע החלפת קוד.
4. בעזרת אובייקט פונקציה, שניתן בפרמטר נוסף לאלגוריתם.