אביב 2011 מועד ב - פתרון

1. **ADT**
2. באיזה קובץ (מבין השלושה) תמצא כל אחת מהשורות הבאות:
   1. struct Set {…...}; **- set.c**
   2. typedef struct Set \* SetP; **- set.h**
   3. struct Date {…..}; **- main.c**
   4. typedef struct Date\* DateP; **- main.c**
   5. typedef void Elem, \*ElemP; **- set.h**

**typedef int (\*CompareElementsFP)(ElemP, ElemP);**

1. השלם את הגדרת הקבוצה

struct Set {

**ElemP\*** elements; //array of elements

int count;

CompareFuncFP compare;

};

4.

SetP intersectSet(SetP set1, SetP set2){

int i1,i2;

int compareRes;

if (set1->compare != set2->compare) return 0;

SetP retVal = createSet(set1->compare);

i1=i2=0;

while ((i1<set1->count) && (i2<set2->count)) {

compareRes=retVal->compare(set1->elements[i1],set2->elements[i2]);

if (!compareRes) {

retVal->elements[retVal->count] = set1->elements[i1];

i1++;

i2++;

retVal->count++;

} else if (compareRes < 0) {

i1++;

} else {

i2++;

}

}

return retVal;

}

5.

int compareDates(ElemP el1, ElemP el2) {

DateP d1= (DateP) el1;

DateP d2= (DateP) el2;

return (d1->year - d2->year)\*500 + (d1->month - d2->month)\*50 + d1->day - d2->day;

}

createSet(**compareDates**);

1. C++

א.

template <class Key, class Value>

class Dictionary

{

private:

List<Key> \_keys;

List<Value> \_values;

Value\* InternalFindVal(const Key&);

const Value \_def;

public:

Dictionary(const Value& def) : \_def(def) { }

**bool AddPair(const Key&, const Value&);**

**Value& GetValue(const Key&) throw(NotFound) /\* the throw list is optional\*/;**

**bool RemovePair(const Key&);**

**Value& operator[](const Key&);**

};

ב.

**...**

**private:**

**List<Key> \_keys;**

**List<Value> \_values;**

**Value\* InternalFindVal(const Key&);**

**...**

**template <class Key, class Value>**

**Value\* Dictionary<Key, Value>::InternalFindVal(const Key& key)**

**{**

**Key\* pKey = \_keys.GetFirst();**

**Value\* pVal = \_values.GetFirst();**

**while (pKey)**

**{**

**if (\*pKey == key)**

**break;**

**pKey = \_keys.GetNext();**

**pVal = \_values.GetNext();**

**}**

**return pVal;**

**};**

ג.

**template <class Key, class Value>**

**Value& Dictionary<Key, Value>::operator[](const Key& key)**

**{**

**Value\* pVal = InternalFindVal(key);**

**if (!pVal) {**

**AddPair(key, \_def);**

**return GetValue(key);**

**}**

**return \*pVal;**

**};**ד.

**class NotFound { };**

**template <class Key, class Value>**

**Value& Dictionary<Key, Value>::GetValue(const Key& key)**

**{**

**Value\* pVal = InternalFindVal(key);**

**if (!pVal)**

**throw NotFound();**

**return \*pVal;**

**};**

1. מה יודפס?

0

v is too small

1

terminate called after throwing an instance of 'bad\_v'

Abort (core dumped)

2

Bad value exception, value= 2

3

terminate called after throwing an instance of 'int'

Abort (core dumped)

4

program terminated

5

6

שאלה 4

BASH

1.

cat \*.[hH] | grep include | grep iostream |wc -l

2.

grep $1 grades | cut –d: -f 5 | cut –d' ' –f3

3.

#!/bin/bash

for i in $1/\*; do

if [[ -f $i ]]; then

head -1 $i

elif [[ -d $i ]]; then

$0 $i

fi

done

הבנה

1. זליגת זכרון מתרחשת כאשר יוצאים מהבלוק ולא משחררים את הזכרון הדינאמי.

במקרה של מצביע חכם, כאשר יוצאים מהבלוק, המצביע החכם קורא לדיסטרקטור שלו ומנקה את הזיכרון אחרי עצמו.

1. מממשים אותו כטמפלט.
2. מאחר ותפקיד המתודה הוא לייצר את האובייקט, ללא שימוש בקונסטרקטור, המתודה נקראת מבלי (וגם לפני) שנוצר אובייקט - חייבים לקרוא למתודה ישירות מהמחלקה.

4.

