**פתרון חורף 2017 מועד ב'**

**פתרון שאלה 1**

**פתרון סעיף א'**

/\*\* ADT of Stack \*/

**typedef** **struct** Stack\_t\* PStack;

/\*User functions\*/

**typedef** PElem (\*CLONE\_FUNC)(PElem);

**typedef** **void** (\*DESTROY\_FUNC)(PElem);

PStack **stackCreate**(CLONE\_FUNC, DESTROY\_FUNC);

**void** **stackDestroy**(PStack stack);

Bool **stackPush**(PStack stack, PElem pElem);

Bool **stackPop**(PStack stack);

PElem **stackTop**(PStack stack);

**int** **stackSize**(PStack stack);

PStack **stackDuplicate**(PStack stack);

**פתרון סעיף 2:**

**void** **stackDestroy**(PStack stack) {

**if** (stack == NULL) {

**return**;

}

Bool result = *FALSE*;

**while** (stackSize(stack) > 0) {

result = stackPop(stack);

assert(result == *TRUE*);

}

**free**(stack);

}

**פתרון סעיף 3:**

Bool **switchStacks**(PStack stackFrom, PStack stackTo)

{

**if**(stackFrom == NULL || stackTo == NULL)

{

**return** *FALSE*;

}

PElem pElem;

**int** size = stackSize(stackFrom);

**int** i = 0;

**for**(i=0;i<size;i++)

{

pElem = stackTop(stackFrom);

stackPush(stackTo, pElem);

stackPop(stackFrom);

}

**return** *TRUE*;

}

**פתרון סעיף 4:**

Bool **queuePop**(PQueue queue) {

**if** (queue == NULL) {

**return** *FALSE*;

}

switchStacks(queue->stackA, queue->stackB);

stackPop(queue->stackB);

switchStacks(queue->stackB, queue->stackA);

**return** *TRUE*;

}

**פתרון סעיף 5:**

PElem **stackDuplicateWrapper**(PElem pElem)

{

**return** stackDuplicate(pElem);

}

**void** **stackDestroyWrapper**(PElem pElem)

{

stackDestroy(pElem);

}

PQueue pQueue = queueCreate(

# **פתרון שאלה 2**

#include<iostream>

using namespace std;

class MultiClassSetItem

{

public:

virtual ~MultiClassSetItem() {};

virtual bool operator==(const MultiClassSetItem& inr) const = 0;

virtual MultiClassSetItem\* dup() const = 0;

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

};

template<class T>

bool operator==(const MultiClassSetItem& inl, const T& inr)

{

const T\* const pinl = dynamic\_cast<const T\* const>(&inl);

if (pinl == NULL)

return false;

return inr == (\*pinl);

}

ostream& operator<<(ostream& os, const MultiClassSetItem& inr)

{

inr.print(os);

return os;

}

class SetNode

{

public:

SetNode(const MultiClassSetItem& data, SetNode\* pNext=NULL) : data(data.dup()), pNext(pNext) {};

~SetNode() { delete data; };

SetNode\* pNext;

MultiClassSetItem\* data;

};

class MultiClassSet

{

public:

MultiClassSet() : head\_(NULL) {}

~MultiClassSet()

{

while (head\_ != NULL)

{

SetNode\* pDel = head\_;

head\_ = head\_->pNext;

delete pDel;

}

}

bool Insert(const MultiClassSetItem& in)

{

SetNode\* pIter = head\_;

while (pIter != NULL)

{

if (\*(pIter->data) == in)

return false;

pIter = pIter->pNext;

}

head\_ = new SetNode(in, head\_);

return true;

}

void print() const

{

SetNode\* pIter = head\_;

cout << "Printing Set:" << endl;

while (pIter != NULL)

{

cout << \*(pIter->data) << endl;

pIter = pIter->pNext;

}

}

private:

SetNode\* head\_;

};

#include<iostream>

#include"MultiClassSet.h"

using namespace std;

class desk : public MultiClassSetItem

{

public:

desk(int width, int length) : width(width), length(length) {};

bool operator==(const desk& inr) const { return inr.width == width && inr.length == length; }

bool operator==(const MultiClassSetItem& inr) const { return inr == (\*this); }

MultiClassSetItem\* dup() const { return new desk(\*this); }

void print(ostream& os) const { os << "desk: " << width << "x" << length; }

private:

int width;

int length;

};

class chair : public MultiClassSetItem

{

public:

chair(int width, int length) : width(width), length(length) {};

bool operator==(const chair& inr) const { return inr.width == width && inr.length == length; }

bool operator==(const MultiClassSetItem& inr) const { return inr == (\*this); }

MultiClassSetItem\* dup() const { return new chair(\*this); }

void print(ostream& os) const { os << "chair: " << width << "x" << length; }

private:

int width;

int length;

};

int main()

{

desk d1(1, 1), d2(3, 4), d3(1, 1), d4(2, 2);

chair c1(2, 2), c2(33, 4), c3(1, 1), c4(2, 2);

MultiClassSetItem\* pd = d1.dup();

delete pd;

if (d1 == d3)

cout << "true" << endl;

MultiClassSet mcs;

mcs.Insert(d1); mcs.Insert(d2); mcs.Insert(d3); mcs.Insert(d4);

mcs.Insert(c1); mcs.Insert(c2); mcs.Insert(c3); mcs.Insert(c4);

mcs.print();

return 0;

}

**פתרון שאלה 3**

**סעיף א'**

פתרון: ישנה זריקת חריגה. צריך לעטוף בבלוק try catch

try {

(ShopMap.at("af54bx63"))->SetFreq(1.6);

(ShopMap.at("xx99xx99"))->SetFreq(2.0);

}

catch(out\_of\_range e){

cerr<< "Erorr out of range " << e.what() <<endl;

}

**סעיף 2**

class compareSP {

public:

bool operator()(Smartphone\* a, Smartphone\* b) {

return a->GetFreq() < b->GetFreq() ;

}

};

int main() {

…

vector<Smartphone\*> ShopVec;

map<const char\*,Smartphone\*>::iterator it;

for ( it = ShopMap.begin(); it != ShopMap.end(); ++it) {

ShopVec.push\_back((it->second));

}

compareSP cSP;

sort(ShopVec.begin(),ShopVec.end(),cSP);

for(int i=0; i<ShopVec.size() ; i++)

ShopVec[i]->print();

…

**סעיף 3**

void print\_if\_cheap(Smartphone\* sp) {

if (sp->isCheap()){

cout<< "Cheap smartphone: " ;

sp->print();

}

}

int main() {

…

for\_each(ShopVec.begin(),ShopVec.end(),print\_if\_cheap);

…

}

**שאלה 4 – פתרון**

**חלק ראשון: BASH (נק')**

1. שורת ה-BASH:

cut -d" " -f3 log | sort | uniq -c | sort -k1 -n | tail -1 |

cut -d" " -f8

בעיקרון uniq -c מקצה 7 תווים למספר, לאחר מכן רווח ולאחר מכן שם הקובץ.

אך, לא מצפים שהסטודנטים ידעו זאת.

1. הסקריפט:

#!/bin/bash

if [[ -d dates ]]; then

rm -rf dates

fi

mkdir dates

dates=`cut -d" " -f1 log | sort | uniq`

for d in $dates; do

grep $d log | cut -d" " -f3 | sort | uniq >> dates/$d

done

**חלק שני: שאלות הבנה (נק')**

בשורה 1 ב- define חסר סוגריים. היה צריך להגדיר אותו כך ((a)+(b)).

בשורה 14 אין בדיקה של ערך החזרה של ה-malloc. יש לבדוק האם מוחזר NULL ולצאת מהפו'.

בשורה 19 תהיה דליפת זיכרון כי לא עשו free(a).

בשורה 21 החזירו מצביע למשתנה מקומי בעוד שהיה צריך לבצע malloc או להחזיר by value.