**אביב 2015 מועד ב פתרון**

**שאלת ADT – פתרון**

**סעיף 1 (8 נק')**

typedef pElement (\*CloneElement) (pElement); // Clone element

typedef void (\*DeleteElement) (pElement); // Delete element

typedef CompResult (\*CompareElements) (pElement, pElement); // Compare two elements

typedef void (\*PrintElement) (pElement); // Print an element

pSortedQueue CreateSortedQueue(CloneElement, DeleteElement, CompareElements, PrintElement);

Result Push(pSortedQueue, pElement);

pElement PopHead(pSortedQueue);

pElement PopTail(pSortedQueue);

Result DeleteSortedQueue(pSortedQueue);

Result PrintSortedQueue(pSortedQueue);

unsigned int GetNumElements(pSortedQueue);

**סעיף 2 (4 נק')**

typedef struct \_sortedQueue {

pNode head;

pNode tail;

unsigned int numElements;

CloneElement cloneElementFunc;

DeleteElement deleteElementFunc;

CompareElements compareElementsFunc;

PrintElement printElementFunc;

} SortedQueue;

**סעיף 3 (3 נק')**

pSortedQueue CreateSortedQueue( CloneElement cloneElementFunc,

DeleteElement deleteElementFunc,

CompareElements compareElementsFunc,

PrintElement printElementFunc )

{

pSortedQueue queue = (pSortedQueue) malloc(sizeof(SortedQueue));

if (!queue) exit(1);

queue->head = NULL;

queue->tail = NULL;

queue->numElements = 0;

queue->cloneElementFunc = cloneElementFunc;

queue->deleteElementFunc = deleteElementFunc;

queue->compareElementsFunc = compareElementsFunc;

queue->printElementFunc = printElementFunc;

return queue;

}

**סעיף 4 (7 נק')**

pElement PopTail(pSortedQueue queue)

{

pElement ret;

pNode t;

if( !queue ) return NULL;

if( queue->numElements == 0 ) return NULL;

if( queue->numElements == 1 )

{

// Removing the last element

// Head and tail points to the same node

ret = queue->head->element;

free( queue->head );

queue->head = NULL;

queue->tail = NULL;

queue->numElements--;

return ret;

}

t = queue->tail;

ret = queue->tail->element;

queue->tail = queue->tail->prev;

queue->tail->next = NULL;

queue->numElements--;

free( t );

return ret;

}

**סעיף 5 (8 נק')**

pElement cloneTask(pElement element)

{

pTask oldTask = (pTask)element;

pTask newTask = (pTask) malloc (sizeof(Task));

if (!newTask) exit(1);

newTask->name = strdup(oldTask->name);

newTask->description = strdup(oldTask->description);

if( (!newTask->name) || (!newTask->description)) exit(1);

newTask->priority = oldTask->priority;

return newTask;

};

void deleteTask(pElement element)

{

pTask task = (pTask)element;

free( task->name );

free( task->description );

free( task );

}

CompResult compareTasks(pElement element1, pElement element2)

{

pTask task1 = (pTask)element1;

pTask task2 = (pTask)element2;

if( task1->priority > task2->priority ) return GREATER;

if( task1->priority < task2->priority ) return LOWER;

return EQUAL;

}

void printTask( pElement element )

{

pTask task = (pTask)element;

if (!task) return;

printf("Task: %s - %s (%d)\n", task->name, task->description, task->priority);

}

**שאלה 2 פתרון**

א.

class GroceryItem {

public:

GroceryItem( string name = "", double price = 0);

virtual GroceryItem& operator+=(double rhs) = 0;

virtual GroceryItem& operator-=(double rhs) = 0;

string GetName() const;

double GetPrice() const;

void SetPrice(double price);

virtual void Print() const = 0;

protected:

string name\_;

double price\_;

};

class PerUnit : public GroceryItem {

public:

PerUnit( string name = "", double price = 0, int amount = 1 );

GroceryItem& operator+=(double rhs);

GroceryItem& operator-=(double rhs);

void Print() const;

protected:

int amount\_;

};

class PerWeight : public GroceryItem {

public:

PerWeight( string name = "", double price = 0, double amount = 1.0);

GroceryItem& operator+=(double rhs);

GroceryItem& operator-=(double rhs);

void Print() const;

protected:

double amount\_;

};

class Alcoholic : public PerUnit {

public:

Alcoholic( string name = "", double price = 0, int amount = 1 );

GroceryItem& operator-=(double rhs);

static void setTradeHr(int StartHr, int EndHr);

protected:

static int StartHr\_;

static int EndHr\_;

};

ב.

GroceryItem& PerUnit::operator-=(double rhs) {

if ( ((int)rhs != rhs) || (rhs < 0) )

throw "Illegal amount";

else if (amount\_ < rhs)

throw "Running out of stock";

else

amount\_ -= (int)rhs;

return \*this;

}

GroceryItem& Alcoholic::operator-=(double rhs) {

int CurrHr = CurrentTime();

if ( (CurrHr < StartHr\_) || (CurrHr >= EndHr\_) )

throw "Illegal trade";

else

return PerUnit::operator-=(rhs);

}

ג.

class Grocery {

public:

bool AddItem( GroceryItem\* item );

bool RemoveItem( string name );

bool AddAmount( string name, double amount );

bool DeleteAmount( string name, double amount );

void Print() const;

protected:

list<GroceryItem\*> inventory\_;

};

bool Grocery::DeleteAmount( string name, double amount ) {

for( list<GroceryItem\*>::iterator it = inventory\_.begin(); it != inventory\_.end(); ++it )

{

if ((\*it)->GetName() == name)

{

try {

\*(\*it) -= amount;

return true;

}

catch (char\* s) {

cout << "Error in " << name << ": " << s << endl;

return false;

}

}

}

cout << "Error in " << name << ": " << "Item doesn't exist" << endl;

return false;

}

float\* cloneCoordinate(float\* coordinate)

{

float\* newCoordinate = (float\*)malloc( sizeof(float) );

\*newCoordinate = \*coordinate;

return newCoordinate;

}

void deleteCoordinate(float\* coordinate)

{

free(coordinate);

}

PPoint PointCreate(int n)

{

PPoint pPoint = (PPoint)malloc( sizeof(Point) );

pPoint->n = n;

pPoint->size = 0;

pPoint->coordinates = ListCreate( cloneCoordinate, deleteCoordinate );

return pPoint;

}

**שאלה 3 – פתרון:**

A:A()

stage 1:

B:B(int)

stage 2:

B:B(int)

C:C(int)

stage 3:

B:B(int)

B:B(int)

D:D()

stage 4:

B:B(B&)

C:print(): num = 1

B:~B()

stage 5:

Error

stage 6:

B:B(int)

B:print(): num = 0

B:op=

stage 7:

Error

stage 8:

B:~B()

stage 9:

//nothing to print from here

B:~B()

B:~B()

C:~C()

B:~B()

B:~B()

A:~A()

**שאלה 4 - BASH, שאלות הבנה** (20 נק')

**חלק א': BASH (10 נק')**

1. (3 נק')

Print\_word

#!bin/bash

len=`echo $1|wc -c`

if [[ $len -lt $2 ]]; then echo $1; fi

1. (7 נק')

find\_train

grep "$1 $2" trains.data | grep @$3 | sort -n -k4 | read\_stations $4

read\_stations

((flag=0))

while read line ; do

split=($line)

if (( ${split[3]} > $1 && !$flag)) ; then

cat "${split[0]}.data"

((flag = 1))

fi

done

if ((!flag)) ; then

echo "No trains were founded"

**חלק ב': שאלות הבנה (10 נק')**

1. (4 נק')
2. כי אם ההקצאה לא הצליחה כל ניסיון לעשות שימוש בfp יגרור שגיאת זמן ריצה (מצביע לזבל)
3. במקרה שnew נכשל בהקצאת זכרון נזרקת חריגה סטנדרטית מסוג bad\_alloc ולכן התכנית תצא (אלא אם כן דאגנו לטפל בזריקה).
4. (3 נק'):

המנגנון שמאפשר זאת , זה האיטרטור. האיטרטור מהווה סוג של מצביע לאלמנטים בתוך הקונטיינרים ובעזרתו הפונקציה שמממשת את האלגוריתם יכולה לגשת לאלמנטים אלה.

1. (3 נק')

הפרמטר מועבר const כי אם לא היה כך אי אפשר היה לגשת לאינדקס בעזרת מספר. השורה

string s4= my\_container[1];

לא תעבור קומפילציה.