# שאלה 1 – ADT (40 נק')

1.

פתרון:

חלק א

1:

typedef struct Question\* PQuestion;

/\*user functions\*/

/\*interface functions\*/

PQuestion QuestCreate(const char\* tit);

void QuestDestroy(PQuestion);

Result QuestAddAnswer(PQuestion, const char\* ans);

void QuestGetHead(PQuestion);

char\* QuestGetNextAns(PQuestion);

2:

struct Question

{

char\* \_quest;

PList \_ans;

};

3:

/\*user functions\*/

char\* cloneAnswers(char\* pAns)

{

char\* nAns = strdup(pAns);

return nAns;

};

void delAnswers(char\* pAns)

{

free(pAns);

};

/\*interface functions\*/

PQuestion QuestCreate(const char\* q)

{

PQuestion pNew = (PQuestion)malloc(sizeof(struct Question));

if (!pNew) return NULL;

pNew->\_ans = ListCreate(cloneAnswers, delAnswers);

pNew->\_quest = strdup(q);

if (!pNew->\_quest) return NULL;

return pNew;

}

חלק ב

(1)

/\*interface functions\*/

PSurvey SurveyCreate(PQuestion);

void SurveyDestroy(PSurvey);

Result SurveyAddAnswer(PSurvey, unsigned int);

double SurveyAnalyzer(PSurvey, analyze\_func);

(2)

struct Survey

{

PQuestion \_quest;

PList \_pAns;

};

חלק ג

(1)

/\*user functions\*/

typedef double (\*analyze\_func)(PList);

(2)

int\* intClone(int\* i)

{

int\* in = (int\*)malloc(sizeof(int));

\*in = \*i;

return in;

};

void intDestroy(int\* i)

{

free(i);

};

double SurveyAnalyzer(PSurvey ps, analyze\_func af)

{

int cur\_ans;

int cur\_count;

PList hist;

char\* curAns;

int\* curChoice;

hist = ListCreate(intClone, intDestroy);

QuestGetHead(ps->\_quest);

cur\_ans = 1;

for(curAns = QuestGetNextAns(ps->\_quest); curAns; curAns = QuestGetNextAns(ps->\_quest)) {

ListGetHead(ps->\_pAns);

cur\_count = 0;

for(curChoice = ListGetNext(ps->\_pAns); curChoice; curChoice = ListGetNext(ps->\_pAns)) {

if (\*((int\*)curChoice) == cur\_ans)

cur\_count = cur\_count + 1;

}

ListAdd(hist, &cur\_count);

cur\_ans = cur\_ans + 1;

}

return af(hist);

}

2. C++ (40 נק)

א

struct node

{

int index;

int value;

node\* next;

};

ב+ג

class arrayList {

public:

arrayList() { head = 0; }

int& operator [](int index)

{

node \*iter =head, \*prev= head, \*temp;

if (head == NULL)

{

head= newnode(index);//insert first in empty list

return head->value;

}

if (head->index > index ) //insert before head

{

temp = newnode(index);

temp->next = head;

head = temp;

return head->value;

}

while ( iter->next != NULL && iter->index < index ){

prev=iter;

iter = iter->next;

}

if (iter->index == index ) //item found

return (iter->value);

temp = newnode(index);

if (iter->next == NULL)

if (iter->index <index)

iter->next =temp;

else

{

temp->next = iter;

prev->next = temp;

}

return (temp->value);

}

friend std::ostream& operator<< (ostream& os, const arrayList& a);

void compress() ;

~arrayList () {

while (head != NULL)

{node\* temp = head;

head = head->next;

delete temp;

}

}

private:

node\* head;

node\* newnode (int index) {

node\* temp;

if ( ( temp = new node ) == NULL ) throw "memory";

temp -> index = index;

temp -> value = 0;

temp -> next = NULL;

return temp;

}

};

ד. מחלקת תור תכיל מערך ואת שני האינדקסים: ראשון ואחרון בתור.

נתחיל תמיד מאינדקס 0.

הכנסת נתון: תכניס לאינדקס הנמוך ביותר -1.

הוצאת נתון: תאפס את האיבר באינדקס האחרון שנכנס, תחסיר 1 מהאינדקס הראשון בתור ואז תפעיל את compress().

class queue {

public:

queue() { lastindex = firstindex = 0);

void insert() (int i)

{

try

{ a[lastindex--] = i; }

catch (char\* s)

{ cout<<"memory"<<endl; }

}

int pop() {

a[firstindex--] = 0;

compress();

}

….

private:

arraylist a;

int firstindex, lastindex;

}

שאלה 3.

חלק א' BASH

א.

#!bin/bash

len=`echo $1|wc -c`

if [[ $len -lt 3 ]]; then echo $1; fi

ב.

#! /bin/bash

ccc=0

for i in $1/\*; do

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

(( ccc=$ccc+1 ))

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

a=`$0 $i`

(( ccc=$ccc + $a ))

fi

done

echo $ccc

ב. UML + DP

1. דיאגרמת מקלות, או ריבוע עם stereotype <<actor>>
2. כדי שלכולן יהיה ממשק משותף, והסובייקט יוכל להכיר רק ממשק אחד ולעבוד בעזרתו עם כל האובזרוורים.
3. כדי שגורם חיצוני לא יוכל ליצור אובייקטים נוספים..