# חורף שאלה 1 - ADT

1. פתרון

typedef struct AdptArray\_

{

int ArrSize;

PElement\* pElemArr;

DEL\_FUNC delFunc;

COPY\_FUNC copyFunc;

}AdptArray;

1. פתרון

PAdptArray CreateAdptArray(COPY\_FUNC copyFunc\_, DEL\_FUNC delFunc\_)

{

PAdptArray pArr = (PAdptArray)malloc(sizeof(AdptArray));

if (pArr == NULL)

return NULL;

pArr->ArrSize = 0;

pArr->pElemArr = NULL;

pArr->delFunc = delFunc\_;

pArr->copyFunc = copyFunc\_;

return pArr;

}

Result SetAdptArrayAt(PAdptArray pArr, int idx, PElement pNewElem)

{

PElement\* newpElemArr;

if (pArr == NULL)

return FAIL;

if (idx >= pArr->ArrSize)

{

// Extend Array

if ((newpElemArr = (PElement\*)calloc((idx + 1), sizeof(PElement))) == NULL)

return FAIL;

memcpy(newpElemArr, pArr->pElemArr, (pArr->ArrSize) \* sizeof(PElement));

free(pArr->pElemArr);

pArr->pElemArr = newpElemArr;

}

// Delete Previous Elem

pArr->delFunc((pArr->pElemArr)[idx]);

(pArr->pElemArr)[idx] = pArr->copyFunc(pNewElem);

// Update Array Size

pArr->ArrSize = (idx >= pArr->ArrSize) ? (idx + 1) : pArr->ArrSize;

return SUCCESS;

}

1. פתרון

void DeleteAdptArray(PAdptArray pArr)

{

int i;

if (pArr == NULL)

return;

for(i = 0; i < pArr->ArrSize; ++i)

{

pArr->delFunc((pArr->pElemArr)[i]);

}

free(pArr->pElemArr);

free(pArr);

}

1. פתרון

PElement GenTreeChildArrCopyFunc(PElement pElem);

void GenTreeChildArrDeleteFunc(PElement pElem);

PGenTree CreateGenTree(PElement pElem\_, COPY\_FUNC copyFunc\_, DEL\_FUNC delFunc\_, PRINT\_FUNC printFunc\_)

{

PGenTree pRoot = (PGenTree)malloc(sizeof(GenTree));

if (pRoot == NULL)

return NULL;

pRoot->fatherChildIdx = -1;

pRoot->pElem = copyFunc\_(pElem\_);

pRoot->pFather = NULL;

pRoot->pChildrenArr = CreateAdptArray(GenTreeChildArrCopyFunc, GenTreeChildArrDeleteFunc);

pRoot->delFunc = delFunc\_;

pRoot->copyFunc = copyFunc\_;

pRoot->printFunc = printFunc\_;

return pRoot;

}

1. פתרון

Result SetChildTreeAt(PGenTree pRoot, int childIdx, PGenTree pChild)

{

Result res;

// Check if Tree or Subtree are Initialized

if (pRoot == NULL || pChild == NULL || pChild->pFather != NULL)

return FAIL;

// Connect Trees

res = SetAdptArrayAt(pRoot->pChildrenArr, childIdx, (PElement)pChild);

if (res == SUCCESS)

{

pChild->pFather = pRoot;

pChild->fatherChildIdx = childIdx;

}

return res;

}

// Implement Internal Functions

PElement GenTreeChildArrCopyFunc(PElement pElem)

{

return pElem;

}

void GenTreeChildArrDeleteFunc(PElement pElem)

{

DeleteGenTree((PGenTree)pElem);

}

1. פתרון

void DeleteGenTree(PGenTree pRoot)

{

PGenTree pFather;

// check if tree exists

if (pRoot == NULL)

return;

// Detach tree from father node (also deletes)

if (pRoot->pFather != NULL)

{

pFather = pRoot->pFather;

pRoot->pFather = NULL;

SetAdptArrayAt(pFather->pChildrenArr, pRoot->fatherChildIdx, NULL);

}

else // This is a true root

{

DeleteAdptArray(pRoot->pChildrenArr);

pRoot->delFunc(pRoot->pElem);

free(pRoot);

}

}

**שאלה 2**

1. הגדרות המחלקות:

class Vertex

{

public:

Vertex(int id, const char\* tag = "");

~Vertex();

double GetOutput();

void ConnectToVertex(Vertex\* vertex, double weight);

virtual void Print(std::ostream& output) const;

virtual void Simulate() = 0;

protected:

std::vector<Vertex\*> m\_connectedVertices;

std::vector<double> m\_weights;

int m\_id;

double m\_output;

char\* m\_tag;

};

class InputVertex :

public Vertex

{

public:

InputVertex(int id, const char\* tag);

~InputVertex();

void PushInput(double value);

virtual void Simulate();

void Print(std::ostream & ro) const;

protected:

std::queue<double> m\_inputs;

};

class ReLUVertex :

public Vertex

{

public:

ReLUVertex(int id, const char\* tag = "", double bias = 0);

~ReLUVertex();

virtual void Simulate();

virtual void Print(std::ostream& ro) const;

protected:

double m\_bias;

};

1. מימוש Simulate:

ב-Vertex אין צורך (וירטואלי טהור).

void InputVertex::Simulate()

{

double value;

if (!m\_inputs.empty())

{

value = m\_inputs.front();

m\_inputs.pop();

}

else

throw "No inputs in queue!";

m\_output = value;

}

void ReLUVertex::Simulate()

{

double sum = 0;

if (m\_connectedVertices.size() == 0)

throw "No Input!";

for (int i = 0; i < m\_connectedVertices.size(); i++)

sum += m\_connectedVertices[i]->GetOutput() \* m\_weights[i];

m\_output = std::max(0.0, sum + m\_bias);

}

1. אובייקט שמעוניין לתפקד כמידע חייב לממש את האופרטורים:

Object::operator+(Object) // for sums  
Object::operator\*(double) // for multiplication by edge weight  
operator<<(ostream, object) // for printing**שאלה 3 – C++ ( 20 נק')**

1. חייבים להיות:
   1. קונסטרקטור דיפולטי
   2. קונסטרקטור העתקה
   3. אופרטור השמה
   4. דיסטרקטור

char\* createNewCopy(const char\* src) {

char\* dst;

if (src==NULL) return NULL;

dst = new char[strlen(src)+1];

strcpy(dst,src);

return dst;

}

class Car {

public:

Car();

Car(int LicensePlate, const char\* Manufacturer);

Car(const Car& rhs);

Car& operator=(const Car& rhs);

~Car();

private:

int LicensePlate\_ ;

char\* Manufacturer\_;

};

Car::Car() : LicensePlate\_(0) , Manufacturer\_(NULL) {}

Car::Car(int LicensePlate, const char\* Manufacturer) : LicensePlate\_(LicensePlate),

Manufacturer\_(createNewCopy(Manufacturer)) {}

Car::~Car() {

if(Manufacturer\_){

delete [] Manufacturer\_;

}

}

Car& Car::operator=(const Car& rhs){

if (this != &rhs) {

LicensePlate\_ =rhs.LicensePlate\_;

if (Manufacturer\_) {

delete [] Manufacturer\_;

}

Manufacturer\_ = createNewCopy(rhs.Manufacturer\_);

}

return \*this;

}

Car::Car(const Car& rhs) :

LicensePlate\_(rhs.LicensePlate\_), Manufacturer\_(createNewCopy(rhs.Manufacturer\_)) {}

1. יש להוסיף לכל אחת מהמחלקות מתודה וירטואלית של הדפסה ובנוסף להגדיר אופרטור >> חיצוני למחלקות(לא חייב להיות friend) שמקבל פרמטר ostream& וconst event&.

char\* createNewCopy(const char\* src) {

char\* dst;

if (src==NULL) return NULL;

dst = new char[strlen(src)+1];

strcpy(dst,src);

return dst;

}

class Event {

public:

Event(const char \* Title, const Date& StartDate);

virtual void print(ostream& os) const;

private:

const char \* Title\_;

Date StartDate\_;

};

Event::Event(const char \* Title, const Date& StartDate):

Title\_(createNewCopy(Title)),StartDate\_(StartDate) {}

void Event::print(ostream& os) const {

os<<Title\_<<" is at "<<StartDate\_.theDay()<<"/"<<StartDate\_.theMonth()<<"/"<<StartDate\_.theYear()<<endl;

}

class Meeting : public Event {

public:

Meeting(const char \* Title, const Date& StartDate, const char \* Location, const char\* WithWho);

virtual void print(ostream& os) const;

private:

char\* Location\_;

char\* WithWho\_;

};

Meeting::Meeting(const char \* Title, const Date& StartDate, const char \* Location, const char\* WithWho) :

Event(Title,StartDate),Location\_(createNewCopy(Location)),WithWho\_(createNewCopy(WithWho)) {}

void Meeting::print(ostream& os) const {

os<<"Meeting: ";

Event::print(os);

os<<"Location: "<<Location\_<<endl;

os<<"With: "<<WithWho\_<<endl;

}

class Vacation : public Event {

public:

Vacation(const char \* Title, const Date& StartDate, const char \* Location,int numDays);

virtual void print(ostream& os) const;

private:

char\* Location\_;

int NumDays\_;

};

Vacation::Vacation(const char \* Title, const Date& StartDate, const char \* Location,int NumDays) :

Event(Title,StartDate),Location\_(createNewCopy(Location)),NumDays\_(NumDays) {}

void Vacation::print(ostream& os) const {

os<<"Vacation: ";

Event::print(os);

os<<"Location: "<<Location\_<<endl;

os<<"Num of days: "<<NumDays\_<<endl;

}

ostream& operator<<(ostream& ro, const Event& event){

event.print(ro);

return ro;

}

**שאלה 4 BASH**

סעיף א':

**average**:

#!/bin/bash  
cat $1 | grep -v "num job" | calc\_average

**calc\_average**:

#! /bin/bash  
(( sum=0 ))  
(( count=0 ))  
while read line; do  
line\_split=($line)  
(( count+=1 ))  
(( sum+=${line\_split[3]}+0 ))  
done  
(( avg=$sum/$count ))  
echo "$avg"

סעיף ב':

**average\_per\_job:**

#!/bin/bash  
jobs=($@)  
(( jobs\_num=$#-1 ))  
for job in ${jobs[@]:1:$jobs\_num}; do  
avg=`grep $job "$1" | calc\_average`  
echo "Average salary of $job: $avg"  
done