Tipuri structurate de date definite de utilizator

Problema 1. Jucarii

Scrieti o aplicatie care il ajuta pe Mos Craciun sa tina evidenta:

- **Jucariilor: idJucarie, denumireJucarie, dimensiuneJucarie**
- **Copiilor: idCopil, numecopil, adresaCopil**
- **Cadourilor: idCadou, idJucarie, idCopil, an**

Si va gestiona o lista de jucarii, o lista de copii si o lista de cadouri cu urmatoarele functionalitati:

- **Adauga, elimina, actualizare jucarie/copil**
- **Cautare jucarie/copil**
- **Creeaza statistici**
    1. **Jucariile cu dimensiunea mai mare decat o dimensiune precizata;**
    2. **Cea mai mica jucarie de un anumit fel (denumire);**
    3. **Toate cadourile pentru un copil dat**
    4. **Toti copiii care primesc aceeasi jucarie data**
    5. **Lista cadourilor dintr-un an dat.**
    6. **Toate jucariile ce sunt livrate la o adresa data.**

Analiza

A. Identificarea entitatilor

- **Jucarie:**
    - **idJucarie: intreg;**
    - **denumireJucarie: string**
    - **dimensiuneJucarie: real;**

- **SirJucarii:**
    - **sir: Jucarie [];**
    - **n: intreg;**

- **Copil:**
    - **idCopil: intreg;**
    - **numeCopil: string**
    - **adresaCopil: string;**

- **SirCopii:**
    - **sir: Copil [];**
    - **n: intreg;**

B. Identificarea si Specificarea subalgoritmilor

Subalgoritmul **citireDinFisier(sJ, sCo, sCa)**:
Descriere: citeste din fisier sirul de jucarii, sirul de copii si sirul de cadouri;
Date: -
Rezultate: sirul de jucarii sJ, sirul de copii sCo, sirul de cadouri sCa;

Sublagoritmul **afisareJucarii(sJ):**
Descriere: afiseaza sirul de jucarii;
Date: un sir de jucarii;
Rezultate: se afiseaza atributele tuturor jucariilor din sir

Sublagoritmul **afisareCopii(sCo):**
Descriere: afiseaza sirul de copii;
Date: un sir de copii;
Rezultate: se afiseaza atributele tuturor copiilor din sir

Sublagoritmul **afisareCadouri(sCa):**
Descriere: afiseaza sirul de cadouri;
Date: un sir de cadouri;
Rezultate: se afiseaza atributele tuturor cadourilor din sir

Functia **listaJucariiDimensiuneMaiMare (jucarii, D):**
Descriere: returneaza o lista cu jucarii a caror dimensiune este mai mare decat cea data;
Date: jucarii ∈ SirJucarii, D: real;
Rezultate: listaJucarii ∈ SirJucarii;
daca (∀ jucarie ∈ jucarii, jucarie.dimensiune > D)
- atunci listaJucarii ’= listaJucarii + jucarie;

Functia **ceaMaiMicaJucarieDenumire (jucarii, denumire):**
Descriere: determina jucaria cu denumriea precizata care are cea mai mica dimensiune;
Date: jucarii ∈ SirJucarii, denumire: string;
Rezultate: jucarie sau NIL
daca (∃ jucarie ∈ jucarii, jucarie.denumire = denumire, jucarie.dimensiune este minima)
- atunci returneaza jucarie
- altfel returneaza NIL;

Sublagoritmul **afiseazaOJucarie(oJucarie):**
Descriere: afiseaza caracteristicile unei jucarii;
Date: o jucarie;
Rezultate: se afiseaza atributele jucariei

Functia **existaCopil(sCo, idCo):**
Descriere: determina daca un copil (identificat prin id) se afla intr-un sir de copii;
Date: un sir de copii si un id de copil
Rezultate: *True*, daca copilul este gasit si *False*, altfel

Functia **toateCadourilePentruUnCopilDat(sC, idCo, sCo):**
Descriere: determina toate cadourile unui copil dat;
Date: un sir de cadouri, un id de copil si un sir de copii;
Rezultate: un sir de cadouri (aferente copilului)

Functia **getCopilCuIdDat(sCo, idCo):**
Descriere: determina dintr-un sir de copii copilul cu un id dat

Date: un sir de copii si un id de copil
Rezultate: copilul cautat

Functia **getJucarieCuIdDat(sJ, idJ):**
Descriere: determina dintr-un sir de jucarii jucaria cu un id dat
Date: un sir de jucarii si un id de jucarie
Rezultate: jucaria cautata

Sublagoritmul **afisareCadouriPentruUnCopil (sC, sJ, sCo):**
Descriere: afiseaza cadourile unui copil
Date: un sir de cadouri, un sir de jucarii, un sir de copii
Rezultate: se afiseaza cadourile unui copil

Functia **existaJucarie(sJ, idJ):**
Descriere: determina daca o jucarie (identificata prin id) se afla intr-un sir de jucarii;
Date: un sir de jucarii si un id de jucarie
Rezultate: *True*, daca cjucaria este gasita si *False*, altfel

Functia **totiCopiiiCarePrimescAceeasiJucarie(sC, idJ, sJ):**
Descriere: determina toate cadourile care contin aceeasi jucarie
Date: un sir de cadouri, un id de jucarie si un sir de jucarii
Rezultate: un sir de cadouri care contin jucaria precizata

Sublagoritmul **afisareCopiiPentruOJucarie (sC, sJ, sCo):**
Descriere: afiseaza copiii care au primit acelasi cadou
Date: un sir de cadouri, un sir de jucarii, un sir de copii
Rezultate: se afiseaza copiii care au primit acelasi cadou

Functia **listaCadourilorDinAnDat (sC, anDat):**
Descriere: determina toate cadourile care au fost oferite intr-un an dat
Date: un sir de cadouri, un an
Rezultate: un sir de cadouri din anul precizat

Sublagoritmul **afisareCadouriPentruAnDat(sC, sJ, sCo):**
Descriere: afiseaza cadourile dintr-un anumit an
Date: un sir de cadouri, un sir de jucarii, un sir de copii
Rezultate: se afiseaza cadourile

Exemple

1. Lista cu jucariilor cu dimensiunea mai amre decat D		
Exemplu	**Date de intrare**	**Date de iesire**
1.	*Sirul de jucarii:* • (1, “minge”, 7.5); • (2, “camion”, 8.5); • (3, “papusa”, 5); • (4, “tamburina”, 15); • (5, “trenulet”, 40); • (6, “cub magic”, 5); • (7, “masina”, 10); • (8, “camion”, 6); • (9, “calut”, 15); • (10, “papusa”, 10). *Dimensiunea D:* 9.00	Lista jucariilor cu dimensiune mai mare decat 9.00 este: • (4, “tamburina”, 15); • (5, “trenulet”, 40); • (7, “masina”, 10); • (9, “calut”, 15); • (10, “papusa”, 10);
2.	*Sirul de jucarii:* • (1, “minge”, 7.5); • (2, “camion”, 8.5); • (3, “papusa”, 5); • (6, “cub magic”, 5); • (8, “camion”, 6); *Dimesiunea D:* 9.00	Lista jucariilor cu dimensiune mai mare decat 9.00 este: Nu exista jucarii cu dimensiunea mai mare decat 9.00.

Implementare C++

```
#include <iostream>
#include <fstream>
#include <string>

typedef struct  {
	int idJ;
	std::string denuJ;
	float dimJ;
} jucarie;

typedef struct {
	int nE;
	jucarie sirE[100];
} sirJucarii;


void citireDinFisier(sirJucarii& sJ, sirCopii& sCo, sirCadouri& sCa) {
	std::string line;
	std::ifstream myfile;
	myfile.open("CadouriMosCraciun.txt");

	if (myfile.is_open())	{
		std::string strNJ;
		myfile >> strNJ;
		int nJ = std::stoi(strNJ);
		sJ.nE = 0;
		for (int i = 0; i < nJ; i++) {
			char strId[20], strDenu[20], strDim[20];
			myfile >> strId >> strDenu >> strDim;
			sJ.sirE[sJ.nE].idJ = std::stoi(strId);
			sJ.sirE[sJ.nE].denuJ = strDenu;
			sJ.sirE[sJ.nE].dimJ = std::stof(strDim);
			sJ.nE++;
		}
		std::string strNCo;
		myfile >> strNCo;
		int nCo = std::stoi(strNCo);
		sCo.nE = 0;
		for (int i = 0; i < nCo; i++) {
				char strIdC[20], strNumeC[20], strAdrC[20];
				myfile >> strIdC >> strNumeC >> strAdrC;
				sCo.sirE[sCo.nE].idC = std::stoi(strIdC);
				sCo.sirE[sCo.nE].numeC=strNumeC;
				sCo.sirE[sCo.nE].adrC= strAdrC;
				sCo.nE++;
			}
		 std::string strNCa;
		 myfile >> strNCa;
		 int nCa =std::stoi(strNCa);
		 sCa.nE = 0;
		for (int i = 0; i < nCa; i++) {
				char strIdCa[20], strIdJ[20], strIdCo[20], strAn[20];
				myfile >> strIdCa >> strIdJ >> strIdCo >> strAn;
				sCa.sirE[sCa.nE].idCa = std::stoi(strIdCa);
				sCa.sirE[sCa.nE].idJ = std::stoi(strIdJ);
				sCa.sirE[sCa.nE].idCo = std::stoi(strIdCo);
				sCa.sirE[sCa.nE].an = std::stoi(strAn);
				sCa.nE++;
			}
		myfile.close();
```

```
	}
	else std::cout << "Unable to open file";

}

void afisareJucarii(sirJucarii sJ) {
	std::cout << "Jucariile lui Mos Craciun sunt: \n";
	for (int i = 0; i < sJ.nE; i++)
	 std::cout<<sJ.sirE[i].idJ<<", "<<sJ.sirE[i].denuJ<<", "<<sJ.sirE[i].dimJ<< "\n";
}
void afiseazaOJucarie(jucarie oJucarie) {
	std::cout << oJucarie.idJ << ", " << oJucarie.denuJ << ", "<<oJucarie.dimJ<<"\n";
}

void afisareCopii(sirCopii sC) {
	std::cout << "Copiii sunt: \n";
	for (int i = 0; i < sC.nE; i++)
	 std::cout<<sC.sirE[i].idC<<", "<<sC.sirE[i].numeC<<", "<<sC.sirE[i].adrC<<"\n";
}

void afisareCadouri(sirCadouri sC) {
	std::cout << "Cadourile sunt: \n";
	std::cout << "idCa "<< ", " << "idJ " << ", " << "idCo "<< ", " << "an "<< "\n";
	for (int i = 0; i < sC.nE; i++)
	 std::cout<<sC.sirE[i].idCa<<", "<<sC.sirE[i].idJ<<", "<<sC.sirE[i].idCo<<
		", "<< sC.sirE[i].an<<"\n";
}

sirJucarii listaJucariiDimensiuneMaiMare(sirJucarii sJ, float dim) {
	sirJucarii sJDim;
	sJDim.nE = 0;
	for (int i = 0; i < sJ.nE; i++)
		if (sJ.sirE[i].dimJ > dim)	{
			sJDim.sirE[sJDim.nE] = sJ.sirE[i];
			sJDim.nE++;
		}
	return sJDim;
}

jucarie ceaMaiMicaJucarieDenumire(sirJucarii sJ, std::string den) {
	jucarie oJucarie;
	oJucarie.denuJ = "";
	oJucarie.idJ = -1;
	oJucarie.dimJ = 1000;
	for (int i = 0; i < sJ.nE; i++)
		if(den==sJ.sirE[i].denuJ)	{
			if (sJ.sirE[i].dimJ < oJucarie.dimJ) {
				oJucarie = sJ.sirE[i];
			}
		}
	return oJucarie;
}

void afiseazaOJucarie(jucarie oJucarie) {
	std::cout<<oJucarie.idJ<<", "<<oJucarie.denuJ<<", "<<oJucarie.dimJ<<"\n";
}

bool existaCopil(sirCopii sCo, int idCo) {
	bool gasit = false;
	int i = 0;
	while ((i < sCo.nE) && (!gasit)) {
```

```
		if (sCo.sirE[i].idC == idCo)
			gasit = true;
		i++;
	}
	return gasit;
}

sirCadouri toateCadourilePentruUnCopilDat(sirCadouri sC, int idCo, sirCopii sCo) {
	sirCadouri sCUnCopil;
	sCUnCopil.nE = 0;
	if (!existaCopil(sCo, idCo))
		std::cout << "Nu exista copilul in lista de copii! \n";
	else{
		for (int i = 0; i < sC.nE;i++)
			if (sC.sirE[i].idCo == idCo)                  {
				sCUnCopil.sirE[sCUnCopil.nE].idCa = sC.sirE[i].idCa;
				sCUnCopil.sirE[sCUnCopil.nE].idCo = sC.sirE[i].idCo;
				sCUnCopil.sirE[sCUnCopil.nE].idJ = sC.sirE[i].idJ;
				sCUnCopil.sirE[sCUnCopil.nE].an = sC.sirE[i].an;
				sCUnCopil.nE++;
			}
	}
	return sCUnCopil;
}

copil getCopilCuIdDat(sirCopii sCo, int idCo) {
	copil c;
	int i = 0;
	bool gasit = false;
	while ((i < sCo.nE) && (!gasit))
		if (sCo.sirE[i].idC == idCo) {
			c = sCo.sirE[i];
			gasit = true;
		}
		else
			i++;
	return c;
}

jucarie getJucariaCuIdDat(sirJucarii sJ, int idJ) {
	jucarie j;
	int i = 0;
	bool gasit = false;
	while ((i < sJ.nE) && (!gasit))
		if (sJ.sirE[i].idJ == idJ) {
			j = sJ.sirE[i];
			gasit = true;
		}
		else
			i++;
	return j;
}

void afisareCadouriPentruUnCopil(sirCadouri sC, sirJucarii sJ, sirCopii sCo) {
	if (sC.nE >= 1) {
		int idCopil = sC.sirE[0].idCo;
		copil unC = getCopilCuIdDat(sCo, idCopil);
		std::cout<<"Copilul "<<unC.numeC<<" a primit urmatoarele cadouri: "<<"\n";
	}
	for (int i = 0; i < sC.nE; i++) {
		int idJucarie = sC.sirE[i].idJ;
```

```
			jucarie oJ = getJucariaCuIdDat(sJ, idJucarie);
			std::cout <<"Jucaria "<< oJ.denuJ<< ", dimensiune "<<oJ.dimJ<<
					", in anul " <<sC.sirE[i].an <<".\n";
		}
	}


	bool existaJucarie(sirJucarii sJ, int idJ) {
		bool gasit = false;
		int i = 0;
		while ((i < sJ.nE) && (!gasit)) {
			if (sJ.sirE[i].idJ == idJ)
				gasit = true;
			i++;
		}
		return gasit;
	}

	sirCadouri totiCopiiiCarePrimescAceeasiJucarie(sirCadouri sC, int idJ, sirJucarii sJ) {
		sirCadouri sCaAceeasiJucarie;
		sCaAceeasiJucarie.nE = 0;
		if (!existaJucarie(sJ, idJ))
			std::cout << "Nu exista jucaria in lista de jucarii! \n";
		else{
			for (int i = 0; i < sC.nE; i++)
			  if (sC.sirE[i].idJ == idJ)     {
				sCaAceeasiJucarie.sirE[sCaAceeasiJucarie.nE].idCa=sC.sirE[i].idCa;
				sCaAceeasiJucarie.sirE[sCaAceeasiJucarie.nE].idCo = sC.sirE[i].idCo;
				sCaAceeasiJucarie.sirE[sCaAceeasiJucarie.nE].idJ = sC.sirE[i].idJ;
				sCaAceeasiJucarie.sirE[sCaAceeasiJucarie.nE].an = sC.sirE[i].an;
				sCaAceeasiJucarie.nE++;
			  }
		}
		return sCaAceeasiJucarie;
	}

	void afisareCopiiPentruOJucarie(sirCadouri sC, sirJucarii sJ, sirCopii sCo) {
		if (sC.nE >= 1) {
			int idJucarie = sC.sirE[0].idJ;
			jucarie oJ = getJucariaCuIdDat(sJ, idJucarie);
			std::cout<<"Jucaria "<<oJ.denuJ << " a fost primita de copiii: " << "\n";
		}
		for (int i = 0; i < sC.nE; i++) {
			int idCopil = sC.sirE[i].idCo;
			copil unC = getCopilCuIdDat(sCo, idCopil);
			std::cout << "Copilul " << unC.numeC << ", adresa " << unC.adrC <<
					", in anul " << sC.sirE[i].an << ".\n";
		}
	}

	sirCadouri listaCadourilorDinAnDat(sirCadouri sC, int anDat) {
		sirCadouri sCaAnDat;
		sCaAnDat.nE = 0;
		for (int i = 0; i < sC.nE; i++)
			if (sC.sirE[i].an == anDat)            {
				sCaAnDat.sirE[sCaAnDat.nE].idCa = sC.sirE[i].idCa;
				sCaAnDat.sirE[sCaAnDat.nE].idCo = sC.sirE[i].idCo;
				sCaAnDat.sirE[sCaAnDat.nE].idJ = sC.sirE[i].idJ;
				sCaAnDat.sirE[sCaAnDat.nE].an = sC.sirE[i].an;
				sCaAnDat.nE++;
			}
```

```
	return sCaAnDat;

}

void afisareCadouriPentruAnDat(sirCadouri sC, sirJucarii sJ, sirCopii sCo) {
	for (int i = 0; i < sC.nE; i++) {
		int idJucarie = sC.sirE[i].idJ;
		jucarie oJ = getJucariaCuIdDat(sJ, idJucarie);
		int idCopil = sC.sirE[i].idCo;
		copil unC = getCopilCuIdDat(sCo, idCopil);
		std::cout <<"Jucaria "<< oJ.denuJ << " cu dim "<<oJ.dimJ<<
			" oferita copilului " << unC.numeC << ", adresa " << unC.adrC <<
			", in anul " << sC.sirE[i].an << ".\n";
	}
}

int main(){
	sirJucarii mySirJucarii;
	sirCopii mySirCopii;
	sirCadouri mySirCadouri;

	citireDinFisier(mySirJucarii, mySirCopii,mySirCadouri);
	afisareJucarii(mySirJucarii);
	afisareCopii(mySirCopii);
	afisareCadouri(mySirCadouri);

	std::cout << "Dati dimensiunea pentru cautare jucarii de dimensiune mai mare: ";
	float dim;
	std::cin >> dim;
	sirJucarii rezSirJucariiDim=listaJucariiDimensiuneMaiMare(mySirJucarii, dim);
	afisareJucarii(rezSirJucariiDim);
	getchar();

	std::cout << "Dati denumirea jucariei: ";
	char denumire[20]; std::cin >> denumire;
	jucarie ceaMaiMicaDenumire = ceaMaiMicaJucarieDenumire(mySirJucarii, denumire);
	afiseazaOJucarie(ceaMaiMicaDenumire);
	getchar();

	//Toate cadourile unui copil dat
	sirCadouri copilSirCadouri;
	int idCopilDat;
	std::cout << "Dati un id copil de determinat toate cadourile: \n";
	std::cin >>idCopilDat;
	copilSirCadouri = toateCadourilePentruUnCopilDat(mySirCadouri,
				idCopilDat, mySirCopii);
	std::cout << "Cadourile pentru copilul cerut: \n";
	afisareCadouriPentruUnCopil(copilSirCadouri, mySirJucarii, mySirCopii);
	getchar();

	//Toti copiii care au primit acelasi cadou
	sirCadouri jucarieSirCadouri;
	int idJucarieData;
	std::cout << "Dati un id jucarie de determinat toti copiii: \n";
	std::cin >> idJucarieData;
	jucarieSirCadouri = totiCopiiiCarePrimescAceeasiJucarie(mySirCadouri,
				idJucarieData, mySirJucarii);
	std::cout << "Copiii pentru jucaria data: \n";
	afisareCopiiPentruOJucarie(jucarieSirCadouri, mySirJucarii, mySirCopii);
	getchar();
```

```
        //Toate cadourile care au fost oferite intr-un an dat
        sirCadouri cadouriAnDat;
        int anDat;
        std::cout << "Dati un an pentru care se doreste lista de jucarii/copii: \n";
        std::cin >> anDat;

        cadouriAnDat = listaCadourilorDinAnDat(mySirCadouri, anDat);
        std::cout << "Cadourile din anul "<< anDat    <<"\n";
        afisareCadouriPentruAnDat(cadouriAnDat, mySirJucarii, mySirCopii);
        getchar();

        getchar();
}
```

Problema 2. Emotii

Scrieti o aplicatie care gestioneaza emotional persoanele dintr-o incapare:

- **Emotie (expresie gura): fericit, trist, surprins, nervos**
- **Persoana care exprima o emotie**
- **Sir de persoane**

Si va gestiona o lista de persoane cu urmatoarele functionalitati:

a) Sa se determine (iterativ si recursiv) din fiecare tip de emotie numarul de persoane care exprima emotia (frecventa emotiei).
b) Sa se determine emotia predominanta din incapere (emotia cu frecventa cea mai mare).
c) Sa se determine toate subsirurile cu o proprietate data ("fericit" are la stanga si la dreapta lui emotie "trist") astfel incat distanta dintre proprietati sa fie mai mica decat o valoare data.

Alte functionalitati (Tema):

d) Sa se determine daca exista persoane vecine care au emotii distincte.

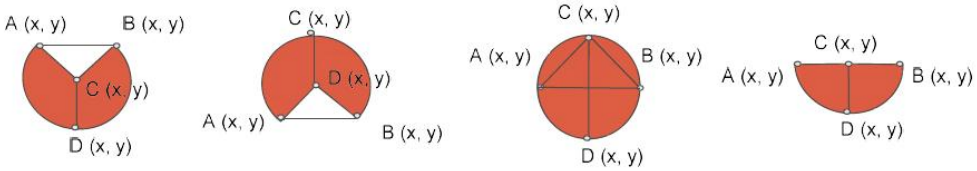
e) Sa se insereze intre oricare doua persoane triste, o persoana vesela (aceeasi persoana).

f) Sa se trimita la terapie persoanele nervoase (eliminare din sir a persoanele nervoase).

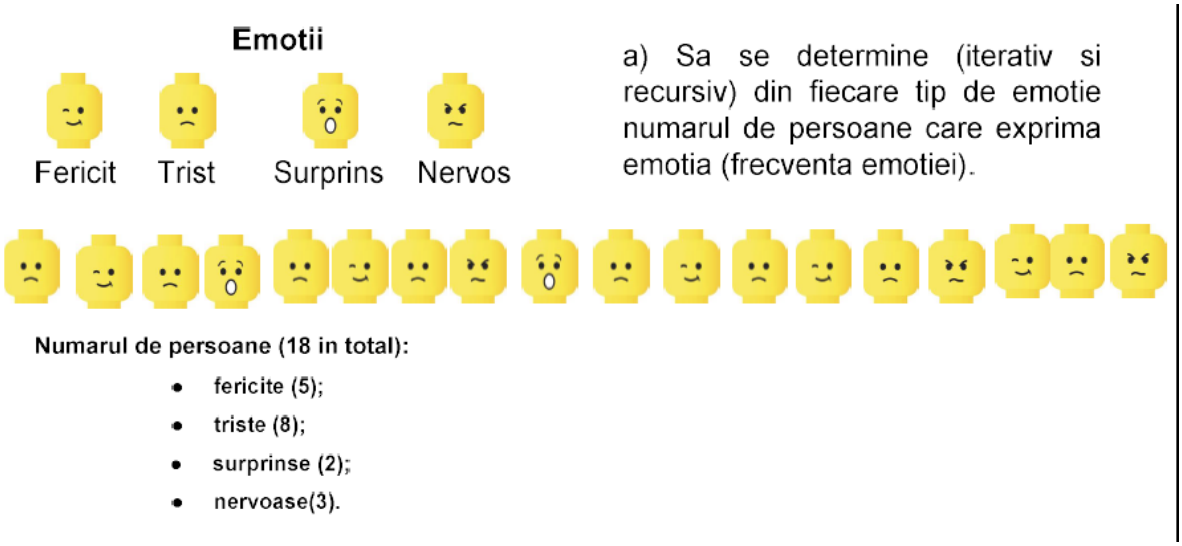
g) Sa se aranjeze persoanele astfel încât cele triste sa nu fie vecine (sau cele nervoase sa nu fie langa cele triste).

Analiza

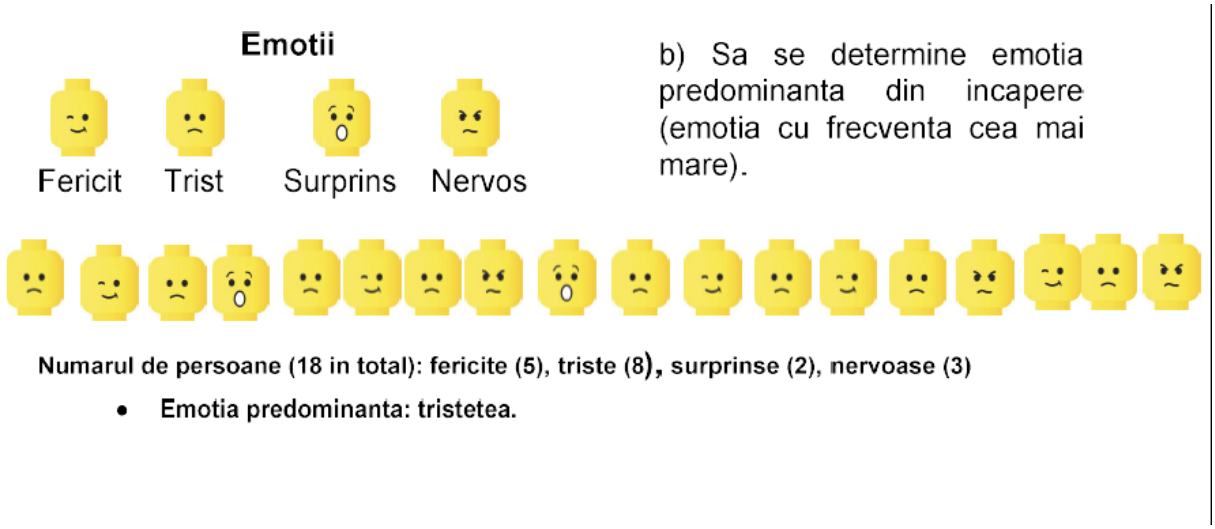
Tipuri de emotii



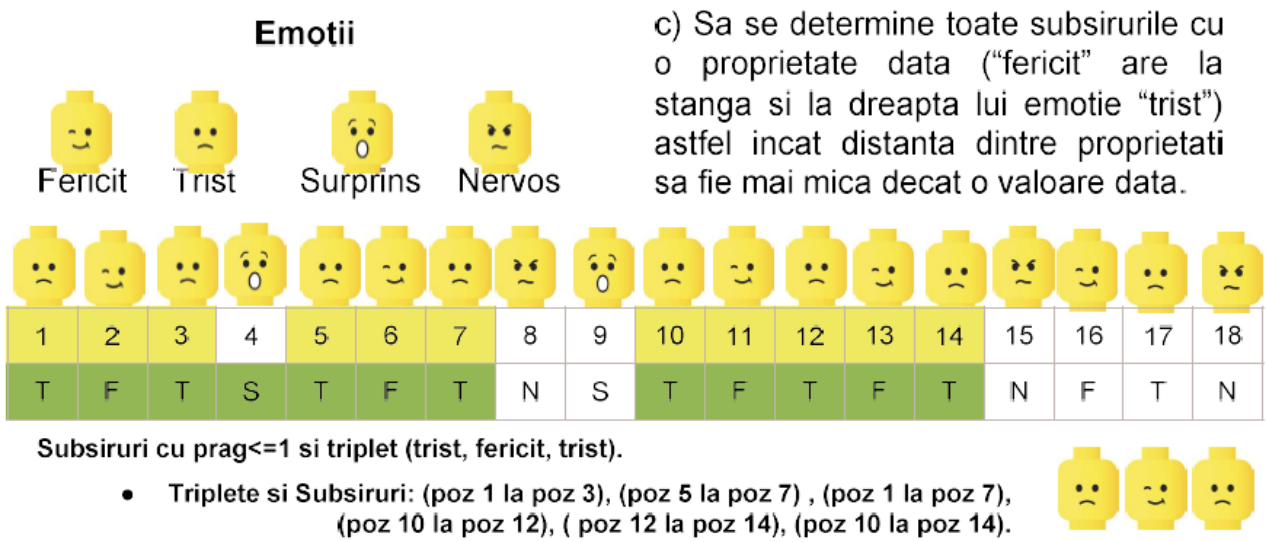
Exemplu pt cerinta a



Exemplu pentru cerinta b



Exemplu pentru cerinta c



Proiectare

A. Identificarea entitatilor:

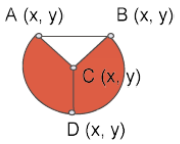
- **Punct:**
  - **x,y: intreg;**
- **Emotie**
  - **A, B, C, D: Punct;**
- **SirEmotii:**
  - **nE:intreg**
  - **sirE:Emotie[ ]**
- **PozSubsir**
  - **pS, pF:intreg**
- **SirPozSubsir**
  - **nE:intreg**
  - **sirPozSubsir: PozSubsir[ ]**

B. Descrierea operatiilor pentru entitati

- citireDinFisierEmotii
- afisareEmotiiPersoane
  - afisareOEmotie
- numarPersoaneFiecareEmotie
  - detEmotieOPersoana
  - emotieFericit
  - emotieSurprins
  - emotieNervos
  - emotieTrist
  - emotieDeBaza
- afisareSubsiruri
  - subsiruriProprietate
  - cautaTriplet
  - adaugaSubSirNou

C. Identificarea si specificarea subalgoritmilor

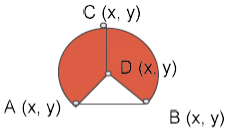
Functia **emotieDeBaza(e):**
Descriere: verifica daca emotia indica o stare valida
Date : o emotie e
Rezultate: true, daca e este o emotie valida, altfel false



Particularitate Emotie Fericit:

- **C.y < A.y**
- **D.y < C.y**

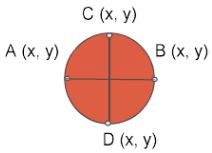
Functia **emotieFericit(e)**
Descriere: verifica daca o emotie indica starea “Fericit”
Date: o emotie e
Rezultate: true, daca (e este o emotie valida si are particularitatile “Fericit”), altfel false



Particularitate Emotie Trist:

- **C.y > A.y**
- **D.y > A.y**
- **D.y < C.y**

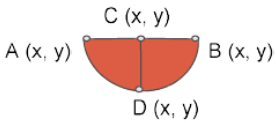
Functia **emotieTrist(e)**
Descriere: verifica daca o emotie indica starea “Trist”
Date: o emotie e
Rezultate: true, daca (e este o emotie valida si are particularitatile “Trist”), altfel false



Particularitate Emotie Surprins:

- **C.y > A.y**
- **D.y < C.y**

Functia **emotieSurprins(e)**
Descriere: verifica daca o emotie indica starea “Surprins”
Date: o emotie e
Rezultate: true, daca (e este o emotie valida si are particularitatile “Surprins”), altfel false



Particularitate Emotie Nervos:

- **C.y = A.y**
- **D.y < C.y**

Functia **emotieNervos(e)**
Descriere: verifica daca o emotie indica starea “Nervos”
Date: o emotie e
Rezultate: true, daca (e este o emotie valida si are particularitatile “Nervos”), altfel false

Subalgoritmul **citireDinFisierEmotii(sEmo)**
Descriere: citeste din fisier un sir de emotii
Date: -
Rezultate : sirul de emotii citit

Functia **numarPersoaneFiecareEmotie(sEmo, tipEmotieDeDeterminat)**
Descriere: : determina numarul de persoane din sir care au o emotie data
Date: un sir de emotii si emotia
Rezultate : numarul persoanelor cu emotia data

Subalgoritmul **determinaEmotiePredominanta(sEmo, maximPredominant, strEmotieDominanta)**
Descriere: `determina emotia predominanta din sirul de emotii`
Date: un sir de emotii sEmo,
Rezultate : frecventa emotiei predominante (maximPredominant) si denumirea acesteia (strEmotieDominanta este din { “fericit”, “trist” “surprins”, “nervos” }); se afiseaza si frecventa fiecarei emotii

Subalgoritmul **cautaTriplet(sEmo; poz; pS; pF)**
Descriere: determina o secventa de emotii incepsi cu pozitia data, care indeplineste proprietatea ceruta “fericit” intre doua persoane “triste”;
Date: sEmo este de tip sirEmotii, pozitie: intreg;
Rezultate: pS este pozitia de inceput, pF este pozitia de sfarsit a unei secvente din emotii care indeplineste proprietatea ceruta (“fericit” intre doua persoane “triste”)

Subalgoritmul **adaugaSubSirNou(mySirPozSubsir; pSNoua; pFNoua);**
Descriere: retine pozitia de inceput si de sfarsit a unui nou subsir care indeplineste proprietatea ceruta “fericit” intre doua persoane “triste”;
Date: mySirPozSubsir ∈ sirPozSubSir, pS, pF: intreg;
Rezultate: mySirPozSubsir’ = mySirPozSubsir + (pS, pF)

Subalgoritmul **subsiruriProprietate(sEmo; prag; mySirPozSubsir);**
Descriere: determina subsirurile care indeplinesc proprietatea “fericit” intre doua persoane “triste” din sirul cu emotii, pentru un prag dat; prag = indica un numar maxim de pozitii acceptat intre doua secvente de emotii determinate, astfel incat acestea sa formeze un singur subsir; in caz contrar secventele formeaza subsiruri distincte;
Date: un sir de emotii sEmo, prag - intreg, subSiruri ∈ SirPozSubSir;
Rezultate: mySirPozSubsir contine toate subsirurile de emotii care respecta proprietatea data, cu pragul maxim dat;

Subalgoritmul **afisareSubsiruri(mySirPozSubsir)**
Descriere: afiseaza toate subsirurile care indeplinesc proprietatea ceruta “fericit” intre doua persoane “triste”, din sirul cu emotii, pentru un prag dat;
Date: emotii este de tip SirEmotii, mySirPozSubsir este de tip sirPozSubSir
Rezultate : se afiseaza subsirurile de emotii care indeplinesc proprietatea si pragul dat

Implementare

```
#include <iostream>
#include <fstream>
#include <string>

typedef struct {
	float x, y;
} punct;

typedef struct {
	punct a,b,c,d;
} emotie;

typedef struct {
	int nE;
	emotie sirE[100];
} sirEmotii;

void citireDinFisierEmotii(sirEmotii& sEmo) {
	std::string line;
	std::ifstream myfile;
	myfile.open("EmotiiPersoane.txt");

	if (myfile.is_open())	{
		std::string strNP;
		myfile >> strNP;
		int nP = std::stoi(strNP);
		sEmo.nE = 0;
		for (int i = 0; i < nP; i++) {
			std::string pctAx, pctAy, pctBx, pctBy, pctCx, pctCy, pctDx, pctDy;
			myfile>>pctAx>>pctAy>>pctBx>> pctBy>> pctCx>> pctCy>> pctDx>> pctDy;
			punct pctA;pctA.x = std::stof(pctAx); pctA.y=std::stof(pctAy);
			punct pctB; pctB.x = std::stof(pctBx); pctB.y= std::stof(pctBy);
			punct pctC; pctC.x = std::stof(pctCx);pctC.y=std::stof(pctCy);
			punct pctD; pctD.x = std::stof(pctDx); pctD.y= std::stof(pctDy);

			sEmo.sirE[sEmo.nE].a = pctA;
			sEmo.sirE[sEmo.nE].b = pctB;
			sEmo.sirE[sEmo.nE].c = pctC;
			sEmo.sirE[sEmo.nE].d = pctD;
			sEmo.nE++;
		}
		myfile.close();
	}
	else std::cout << "Unable to open file";

}

int numarPersoaneFiecareEmotie(sirEmotii sEmo,std:: string tipEmotieDeDeterminat){
	int nrPersoaneOEmotie = 0;
	for (int i = 0; i < sEmo.nE; i++) {
		std::string tipEmotie = detEmotieOPersoana(sEmo.sirE[i]);
		if (tipEmotie ==  tipEmotieDeDeterminat) {
			nrPersoaneOEmotie++;
		}
	}
	return nrPersoaneOEmotie;
}
```

```
int numarPersoaneFiecareEmotieRecursiv(sirEmotii sEmo, std::string
tipEmotieDeDeterminat, int i) {
	if (i == -1)
		return 0;
	else{
		std::string tipEmotie;
		tipEmotie = detEmotieOPersoana(sEmo.sirE[i]);
		if (tipEmotie == tipEmotieDeDeterminat)
			return 1 + numarPersoaneFiecareEmotieRecursiv(sEmo,
					tipEmotieDeDeterminat, i - 1);
		else
			return numarPersoaneFiecareEmotieRecursiv(sEmo,
					tipEmotieDeDeterminat, i - 1);
	}
}

void determinaEmotiePredominanta(sirEmotii sEmo,int& maximPredominant,  std::string
strEmotieDominanta) {
	int nrFericite = 0;
	nrFericite = numarPersoaneFiecareEmotie(sEmo, "fericit");
	std::cout << "Numar de persoane fericite: " << nrFericite;
	getchar();

	//RECURSIV
	int nrFericiteRec = 0;
	int i = sEmo.nE;
	nrFericiteRec = numarPersoaneFiecareEmotieRecursiv(sEmo, "fericit", i - 1);
	std::cout << "Numar de persoane fericite (***RECURSIV***): " << nrFericiteRec;
	getchar();

	int nrTriste = 0;
	nrTriste = numarPersoaneFiecareEmotie(sEmo, "trist");
	std::cout << "Numar de persoane triste: " << nrTriste;
	getchar();

	int nrSurprinse = 0;
	nrSurprinse = numarPersoaneFiecareEmotie(sEmo, "surprins");
	std::cout << "Numar de persoane surprins: " << nrSurprinse;
	getchar();

	int nrNervoase = 0;
	nrNervoase = numarPersoaneFiecareEmotie(sEmo, "nervos");
	std::cout << "Numar de persoane nevoase: " << nrNervoase;
	getchar();

	strEmotieDominanta = "fericit";

	if (maximPredominant < nrTriste) {
		maximPredominant = nrTriste;
		strEmotieDominanta = "trist";
	}
	else
		if (maximPredominant < nrSurprinse) {
			maximPredominant = nrSurprinse;
			strEmotieDominanta = "surprins";
		}
		else
			if (maximPredominant < nrNervoase) {
				maximPredominant = nrNervoase;
				strEmotieDominanta = "nervos";
			}
```

```
}

bool emotieTrist(emotie e) {
	bool eT = false;
	if ((emotieDeBaza(e))&&(e.c.y > e.a.y)&&(e.d.y>e.a.y) && (e.d.y < e.c.y))
		eT = true;
	return eT;
}

bool emotieFericit(emotie e) {
	bool eF = false;
	if ((emotieDeBaza(e)) && (e.c.y < e.a.y) && (e.d.y < e.c.y))
		eF = true;
	return eF;
}

void cautaTriplet(sirEmotii sEmo, int poz, int& pS, int& pF) {
	pS = -1; pF = -1; //cod de eroare cand nu gasesc triplet
	bool gasitT = false;
	while ((!gasitT) &&(poz<sEmo.nE)) {
		while ((poz<sEmo.nE)&&(!emotieTrist(sEmo.sirE[poz])))
			poz++;		//daca am gasit pozitie trist si mai am cel putin
					//2 pozitii de verificat
		if ((poz < sEmo.nE) && ((poz + 2) < sEmo.nE)) {
			if (emotieFericit(sEmo.sirE[poz + 1]) &&
					(emotieTrist(sEmo.sirE[poz + 2]))){
				pS = poz;
				pF = poz + 2;
				gasitT = true;
			}
			else
				poz++;
		}
		else
			poz++;
	}//while gasitT
}

void adaugaSubSirNou(sirPozSubsir& mySirPozSubsir, int pSNoua,int pFNoua){
	mySirPozSubsir.sirSubsir[mySirPozSubsir.nE].pS = pSNoua;
	mySirPozSubsir.sirSubsir[mySirPozSubsir.nE].pF = pFNoua;
	mySirPozSubsir.nE++;
}

sirPozSubsir subsiruriProprietate(sirEmotii sEmo, int prag, sirPozSubsir&
mySirPozSubsir){
	mySirPozSubsir.nE = 0;
	int pS=-1, pF=-1, pS2=-1, pF2=-1;
	bool existaTriplet2 = false;
	int i = 0;
	cautaTriplet(sEmo, i, pS, pF);
	adaugaSubSirNou(mySirPozSubsir, pS, pF);

	if ((pS != -1) && (pF != -1)) {
		cautaTriplet(sEmo, pF, pS2, pF2); //caut incepand de la pF
		adaugaSubSirNou(mySirPozSubsir, pS2, pF2);
		if ((pS2 != -1) && (pF2 != -1))
			existaTriplet2 = true;
		while ((existaTriplet2)){
			if ((pS2 - pF-1) <= prag) {
```

```
                        //distanta <=prag ==> subsecventa mai lunga devine cea
                        //contopita din secv1 si secv2
                        pS = pS;
                        pF = pF2;
                        adaugaSubSirNou(mySirPozSubsir, pS, pF);
                  }
                  else{  //prag prea mare ==> prima subsecenta devine a doua
                         //subsecventa  si continuu cautare de dupa secventa 2
                        pS = pS2;
                        pF = pF2;
                        adaugaSubSirNou(mySirPozSubsir, pS, pF);
                  }
                  cautaTriplet(sEmo, pF, pS2, pF2); //caut incepand de la pF
                                                    // (poate fi 'trist')
                  if ((pS2 != -1) && (pF2 != -1)) {
                        existaTriplet2 = true;
                        adaugaSubSirNou(mySirPozSubsir, pS2, pF2);
                  }
                  else
                        existaTriplet2 = false;
            }//while existaTriplet2
      }//a fost gasit primul triplet
      /*else
      {
            std::cout << "Nu sunt subsiruri in sir cu propritatea ceruta. \n";
      }*/
      return mySirPozSubsir;
}

void afisareSubsiruri(sirPozSubsir mySirPozSubsir) {
      for (int j = 0; j < mySirPozSubsir.nE; j++)
            std::cout << mySirPozSubsir.sirSubsir[j].pS << " , " <<
                                    mySirPozSubsir.sirSubsir[j].pF << "\n";

}

int main() {
      sirEmotii mySirEmo;
      citireDinFisierEmotii(mySirEmo);
      int maximDominat = -1;
      std::string strEmotieDominanta = "";
      determinaEmotiePredominanta(mySirEmo, maximDominat, strEmotieDominanta);
      std::cout << "Emotia predominanta este:" << strEmotieDominanta <<
                  " cu numar maxim " << maximDominat << "\n";
      int prag = 1;
      sirPozSubsir mySirPozSubsir;
      subsiruriProprietate(mySirEmo, prag, mySirPozSubsir);
      afisareSubsiruri(mySirPozSubsir);
      getchar();
}
```

Exemple

22 1 6 3 6 2 8 2 7 1 7 3 7 2 6 2 5 1 6 3 6 2 8 2 7 1 6 3 6 2 7 2 5 1 6 3 6 2 8 2 7 1 7 3 7 2 6 2 5 1 6 3 6 2 8 2 7 1 7 3 7 2 6 2 5 1 6 3 6 2 8 2 7 1 7 3 7 2 6 2 5 1 6 3 6 2 7 2 5 1 6 3 6 2 8 2 7 1 6 3 6 2 8 2 7 1 7 3 7 2 6 2 5 1 6 3 6 2 8 2 7 1 6 3 6 2 6 2 5 1 6 3 6 2 8 2 7 1 7 3 7 2 6 2 5 1 6 3 6 2 8 2 7 1 6 3 6 2 8 2 7 1 7 3 7 2 6 2 5 1 6 3 6 2 8 2 7	a) trist: 12 fericit: 7 surprins: 2 nervos: 1 b) emotia predominanta: trist frecventa = 12 c) 2 subsiruri 1 6 3 6 2 8 2 7 1 7 3 7 2 6 2 5 1 6 3 6 2 8 2 7 1 6 3 6 2 7 2 5 1 6 3 6 2 8 2 7 1 7 3 7 2 6 2 5 1 6 3 6 2 8 2 7 1 7 3 7 2 6 2 5 1 6 3 6 2 8 2 7 Si 1 6 3 6 2 8 2 7 1 7 3 7 2 6 2 5 1 6 3 6 2 8 2 7 1 6 3 6 2 6 2 5 1 6 3 6 2 8 2 7 1 7 3 7 2 6 2 5 1 6 3 6 2 8 2 7 1 6 3 6 2 8 2 7 1 7 3 7 2 6 2 5 1 6 3 6 2 8 2 7