1. Да се дефинира класата Pesna во која се чуваат информации за (5 поени):

* име на песна (динамички алоцирано поле од знаци)
* времетраење на песната во минути
* тип на песна кој може да биде: поп, рап или рок (енумерација tip)

Да се обезбедат сите потребни методи за класата за правилно функционирање на програмата. (10 поени)

Дополнително за оваа класа да се дефинира методот:

* pecati() - функција во која ќе се печатат одделени со цртичка(-): името на песната во наводници и времетраењето на песната со постфикс min. (5 поени)

Да се креира класата CD во која се чуваат информации за (5 поени):

* низа од песни снимени на CD-то (Поле од најмногу 10 објекти од класата Pesna )
* број на песни снимени на CD-то
* максимално времетраење на песните на CD-то (во минути)

Да се обезбедат сите потребни методи за класата за правилно функционирање на програмата. (5 поени)

Дополнително за оваа класа да се дефинираат методите:

* dodadiPesna (Pesna p) - со кој се додава песната p на CDто, но само ако има доволно преостанато место (земајќи го предвид максималното времетраење на песните на CDто и времетраењата на песните на CD-то) и ако има помалку од 10 песни на CD-то. (10 поени)
* pecatiPesniPoTip(tip t) – со кој се печатат информациите за сите песни од тип t (поп, рап или рок). (10 поени)

|  |  |
| --- | --- |
| INPUT | EXPECTED |
| 3  4  Happy 5 0  Layla 8 2  Stan 4 1  Revolution 7 2 | ===== Testiranje na metodot dodadiPesna() od klasata CD ======  "Happy"-5min  "Layla"-8min  "Stan"-4min |
| 2  4  Happy 5 0  Layla 2 2  Stan 2 1  Revolution 3 2 | ===== Testiranje na klasata CD ======  "Happy"-5min  "Layla"-2min  "Stan"-2min  "Revolution"-3min |
| 5  4  Happy 5 0  Layla 8 2  Stan 4 1  Revolution 7 2  2 | ===== Testiranje na metodot pecatiPesniPoTip() od klasata CD ======  "Layla"-8min |
| 5  11  Happy 1 0  Layla 1 2  Stan 1 1  Revolution 1 2  Love 1 0  Life 1 0  Photograph 1 2  Listen 1 0  Anaconda 1 1  Game 1 1  Hello 1 0  0 | ===== Testiranje na metodot pecatiPesniPoTip() od klasata CD ======  "Happy"-1min  "Love"-1min  "Life"-1min  "Listen"-1min |
| 5  4  Happy 5 0  Layla 8 2  Stan 4 1  Revolution 7 2  1 | ===== Testiranje na metodot pecatiPesniPoTip() od klasata CD ======  "Stan"-4min |
| 1  Happy 5 0 | ===== Testiranje na klasata Pesna ======  "Happy"-5min |
| 4  4  Happy 5 0  Layla 2 2  Stan 2 1  Revolution 3 2  2 | ===== Testiranje na metodot pecatiPesniPoTip() od klasata CD ======  "Layla"-2min  "Revolution"-3min |

2. Да се имплементира класа List во којашто ќе се чуваат информации за:

* броеви што се дел од листата (динамички алоцирана низа од цели броеви)
* бројот на броеви што се дел од листата

За класата да се дефинираат следните методи:

* конструктор (со аргументи), copy-конструктор, деструктор, оператор =
* void pecati()метод што ќе ги печати информациите за листата во форматот: [број на елементи во листата]: x1 x2 .. xn sum: [сума] average: [просек]
* int sum() метод што ја враќа сумата на елементите во листата
* double average() метод што ќе го враќа просекот на броевите во листата

Дополнително, креирајте класата ListContainer, во која што ќе се чуваат информации за:

* низа од листи (динамички алоцирана низа од објекти од класата List)
* број на елементи во низата од листи (цел број)
* број на обиди за додавање на листа во контејнерот (цел број првично поставен на нула)

За класата потребно е да ги дефинирате следните методи:

* конструктор (default), copy-конструктор, деструктор, оператор =
* void pecati() метод што ќе ги печати информациите за контејнерот во форматот: List number: [реден број на листата] List info: [испечатени информации за листата со методот List::pecati()] \n sum: [сума] average: [просек]
  + доколку контејнерот е празен се печати само порака The list is empty.
* void addNewList(List l) метод со којшто се додава листа во контејнерот
  + **Напомена**: една листа се додава во контејнерот ако и само ако има различна сума од сите листи што се веќе додадени во контејнерот
* int sum() метод што ја враќа сумата на сите елементи во сите листи
* double average() метод што го враќа просекот на сите елементи во сите листи во контејнерот

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| 2  1  1  2  0 1  0 | List number: 1 List info: 1: 1 sum: 1 average: 1  Sum: 1 Average: 1  Successful attempts: 1 Failed attempts:1 |
| 5  5  1 2 3 4 5  2  10 5  3  1 2 3  6  1 4 5 6 7 10  3  10 10 13  0 | List number: 1 List info: 5: 1 2 3 4 5 sum: 15 average: 3  List number: 2 List info: 3: 1 2 3 sum: 6 average: 2  List number: 3 List info: 6: 1 4 5 6 7 10 sum: 33 average: 5.5  Sum: 54 Average: 3.85714  Successful attempts: 3 Failed attempts:2 |
| 3  2  1 2  1  3  3  1 1 1  0 | List number: 1 List info: 2: 1 2 sum: 3 average: 1.5  Sum: 3 Average: 1.5  Successful attempts: 1 Failed attempts:2 |
| 6  5  1 2 3 4 5  2  1 2  3  1 1 1  4  1 1 1 1  3  1 1 2  5  10 10 10 10 10  0 | List number: 1 List info: 5: 1 2 3 4 5 sum: 15 average: 3  List number: 2 List info: 2: 1 2 sum: 3 average: 1.5  List number: 3 List info: 4: 1 1 1 1 sum: 4 average: 1  List number: 4 List info: 5: 10 10 10 10 10 sum: 50 average: 10  Sum: 72 Average: 4.5  Successful attempts: 4 Failed attempts:2 |
| 5  5  1 2 3 4 5  2  10 5  3  1 2 3  6  1 4 5 6 7 10  3  10 10 13  1 | Test case for operator =  The list is empty  0 54  List number: 1 List info: 5: 1 2 3 4 5 sum: 15 average: 3  List number: 2 List info: 3: 1 2 3 sum: 6 average: 2  List number: 3 List info: 6: 1 4 5 6 7 10 sum: 33 average: 5.5  Sum: 54 Average: 3.85714  Successful attempts: 3 Failed attempts:2  54 54 |

3. За потребите на еден приватен банкар кој тргува со акции, потребно е да се направи модул за менаџирање со клиентски портфолија.

За таа цел, треба да се креира класа StockRecord. Класата треба да содржи:

* ID, кој означува која е акцијата која се тргува (низа од 12 знаци)
* Име на компанијата која ја издала акцијата (низа од 50 знаци)
* Цена според која се купени акциите (реален број)
* Моментална цена на акцијата (реален број)
* Број на акции кои се купени (цел број)

За класата да се дефинираат потребните конструктори и *set* метода за моменталната цена со потпис void setNewPrice(double c) (**10 поени**). Исто така, да се имплементираат методи double value() и double profit(). Првата треба да ја пресметува вредноста на акциите, а втората треба да го пресметува моменталниот профит од акциите. Пресметката се врши на следниот начин (**5 поени**):

Вредност = број на акции \* моментална цена

Профит = број на акции \* (моментална цена - цена по која се купени)

Дополнително, да се преоптовари операторот << за печатење на објектот во следниот формат (**5 поени**): *Компанија БројНаАкции КуповнаЦена МоменталнаЦена Профит*

Да се креира класа Client која содржи:

* Име и презиме на клиентот (низа од 60 знаци)
* ID на сметка (цел број)
* Низа од компании кои ги поседува клиентот (динамичка низа од објекти од класата StockRecord)
* Број на објекти во низата (цел број)

За класата Client да се дефинираат потребните конструктори, деструктор и метода double totalValue() која ќе ја пресметува моменталната вредност на акциите кои ги поседува клиентот. Таа се пресметува како збир од вредноста на сите акции од секоја компанија од кои е составено клиентското портфолио (**15 поени**). Дополнително, за класата да се преоптоварат операторите:

+= за купување на нова компанија во портфолиото (додавање на нов објект од класата StockRecord во низата со компании) (**10 поени**)

<< за печатење на информации за портфолиото на клиентот така што во првиот ред ќе бидат прикажани ID на сметката и моменталната вредност на портфолиото одделени со празно место, а во секој нареден ред ќе бидат прикажани компаниите од кои е составено портфолиото, секоја во посебен ред (**5 поени**)

Комплетна функционанлост (**5 поени**)

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| 3  Aleksandar  240992  3  CH0038863350  Nestle  100  90  100  DE000BAY0017  Bayer  60  70  500  DE0007164600  SAP  110  115  200 | =====TEST NA KLASATA Client=====  Konstruktor OK  Operator += OK  240992 67000  Nestle 100 100 90 -1000  Bayer 500 60 70 5000  SAP 200 110 115 1000  Operator << OK |
| 2  2  DE0008404005  Allianz  120  100  100  US5949181045  Microsoft  60  90  100 | =====TEST NA METODITE I OPERATOR << OD KLASATA StockRecord=====  10000  Allianz 100 120 100 -2000  9000  Microsoft 100 60 90 3000 |
| 1  90 | =====TEST NA KLASATA StockRecord=====  Konstruktor OK  SET metoda OK |

4. Да се креира класа Patnik во која се чуваат податоци за патниците на еден воз и тоа: име (низа од максимум 100 знаци), класа на вагон во која се вози (цел број 1 или 2) и велосипед (булова променлива).

За оваа класа да се преоптоварат:

* Оператор << - за печатење на информациите за патникот во формат:

Име на патник

Бројот на класата (1 или 2)

Дали има велосипед?

Потоа да се креира клaса Voz во која се чува: крајна дестинација (низа од максимум 100 знаци), динамички алоцирана низа од објекти од класата Patnik, како и број на елементи во низата (цел број), број на дозволени велосипеди (цел број).

За класата да се обезбедат:

* Оператор += – за додавање нов патник во воз. Патник со велосипед ќе може да се качи во воз само ако возот го дозволува тоа.
* Оператор << - за печатење на крајната дестинација до која вози и листата на патници
* Функција со потпис: patniciNemaMesto(). Со оваа функција се пресметува вкупниот број на патници (од 1-ва класа и 2-ра класа) за кои нема место да влезат во возот со велосипед. Притоа треба да се внимава дека во пресметувањето на вкупниот број патници со велосипеди кои ќе влезат во возот прво треба да се земат предвид патниците од 1-ва класа, а потоа се дозволува да влегуваат патниците од 2-ра класа се додека не се достигне максималниот број на дозволени велосипеди во возот. На крај се печати бројот на патници со велосипеди кои остануваат (не влегуваат во возот) од 1-ва класа, а потоа од 2-ра класа.

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| Amsterdam  0  5  Ana  1  1  Kate  2  0  Ace  2  0  Borce  2  1  Mimi  2  0 | Amsterdam  Kate  2  0  Ace  2  0  Mimi  2  0  Brojot na patnici od 1-va klasa koi ostanale bez mesto e: 0  Brojot na patnici od 2-ra klasa koi ostanale bez mesto e: 0 |
| Dizeldorf  10  3  Ana  1  1  Kate  2  0  Ace  2  0 | Dizeldorf  Ana  1  1  Kate  2  0  Ace  2  0  Brojot na patnici od 1-va klasa koi ostanale bez mesto e: 0  Brojot na patnici od 2-ra klasa koi ostanale bez mesto e: 0 |

5. За потребите на една продавница за гитари, потребно е да се направи програма за менаџирање на магацинот.

За таа цел, треба да се креира класа Gitara **(5 поени)**. Класата треба да содржи:

* Сериски број (низа од 25 знаци)
* Набавна цена (реален број)
* Година на производство (цел број)
* Тип (низа од 40 знаци)

Да се дефинира метод daliIsti кој како аргумент ќе добие објект од тип Gitara и ќе ги спореди двете гитари според нивниот сериски број **(5 поени)**

Дополнително, за оваа класа да се направи функција pecati() за печатење на објектот во следниот формат **(5 поени)**:

*[СерискиБрој] [Тип] [Цена]*

Да се креира класа Magacin **(5 поени)** која содржи:

* Име на магацинот (низа од 30 знаци)
* Локација на магацинот (низа од 60 знаци)
* Низа од гитари кои ги поседува магацинот (динамички алоцирана низа од објекти од класата Gitara)
* Број на објекти во низата (цел број)
* Година на отвoрање (цел број)

За класата Magacin да се дефинира метод double vrednost() што ќе ја пресметува моменталната вредност на гитарите кои се наоѓаат во магацинот. Таа се пресметува како збир од вредноста на сите гитари од кои е составен магацинот **(5 поени)**.

Дополнително, за класата да се дефинираат функциите:

-void dodadi(Gitara d) за сместување на нова гитара во магацин (додавање на нов објект од класата Gitara во динамички алоцираната низа со гитари) **(10 поени)**

-void prodadi(Gitara p) за бришење на дадена гитара од магацинот (бришење на сите објекти кои се исти како аргументот проследен во фунцијата)**(10 поени)**

-void pecati(bool daliNovi) за печатење на информации за магацинот така што во првиот ред ќе биде прикажано имете и локацијата на магацинот, а потоа во секој нареден ред ќе бидат прикажани гитарите кои се наоѓаат во магацинот. Доколку пратената променлива има вредност true ќе се печатат само гитарите кои имаат година на производство понова од годината на отварање на магацинот. Во спротивно се печатат сите гитари **(5 поени)**.

За класите да се дефинираат потребните конструктори, деструктор и сите останати методи за правилно функционирање на програмата **(5 поени)**. Сите податочни членови во класите се приватни.

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| 2 | ===== Testiranje na klasata Magacin so metodot print() ======  Magacin1 Lokacija1 |
| 1 Acoustic 55333 2012 33212.40 | ===== Testiranje na klasata Gitara ======  Acoustic  55333  2012  33212.4 |
| 5  5  Acoustic1  55113  2016  35212.9  Electrical  111111  2015  11212.8  Acoustic2  55113  2015  35212.9  Acoustic3  55113  2010  35212.9  Electrical2  333333  2010  11212.8 | ===== Testiranje na klasata Magacin so metodot prodadi() i pecati(true) ======  gitara dodadi  gitara dodadi  gitara dodadi  gitara dodadi  gitara dodadi  Magacin1 Lokacija1  55113 Acoustic1 35212.9  111111 Electrical 11212.8  55113 Acoustic2 35212.9  Po brisenje:  Magacin1 Lokacija1  111111 Electrical 11212.8 |
| 6  5  Acoustic1  55113  2016  35212.9  Electrical  111111  2015  11212.8  Acoustic2  55113  2015  35212.9  Acoustic3  55113  2010  35212.9  Electrical2  333333  2010  11212.8 | ===== Testiranje na klasata Magacin so metodot vrednost()======  128064  Po brisenje:  22425.6 |
| 4  5  Acoustic1  55113  2010  35212.9  Electrical  111111  2012  11212.8  Acoustic2  55113  2010  35212.9  Acoustic3  55113  2010  35212.9  Electrical2  333333  2015  11212.8 | ===== Testiranje na klasata Magacin so metodot prodadi() ======  gitara dodadi  gitara dodadi  gitara dodadi  gitara dodadi  gitara dodadi  Magacin1 Lokacija1  55113 Acoustic1 35212.9  111111 Electrical 11212.8  55113 Acoustic2 35212.9  55113 Acoustic3 35212.9  333333 Electrical2 11212.8  Magacin1 Lokacija1  111111 Electrical 11212.8  333333 Electrical2 11212.8 |
| 3  2  Acoustic  55113  2010  35212.9  Electrical  33113  2011  11212.8 | ===== Testiranje na klasata Magacin so metodot dodadi() ======  gitara dodadi  gitara dodadi  Magacin1 Lokacija1  55113 Acoustic 35212.9  33113 Electrical 11212.8 |

6. Да се креира структура Воз во која се чуваат податоци за релацијата по која се движи возот (низа од најмногу 50 знаци), предвиден бројот на километри што треба да се поминат (реален број), како и бројот на патници во возот (цел број).

Потоа да се креирa структура ZeleznickaStanica во која се чуваат податоци за градот во кој се наоѓа (низа од 20 знаци), низа од возови што поаѓаат од станицата (најмногу 30) и бројот на возови (цел број).

Треба да се направи функција со потпис

void najkratkaRelacija(ZeleznickaStanica\* zs, int n, char\* grad)

во која се печати релацијата и бројот на километри на возот што поминува најкратка релација (најмалку километри), а поаѓа од железничката станица од градот што се проследува како влезен аргумент. Ако има повеќе возови со ист најмал број на километри, да се испечати релацијата на последниот таков.

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| 1  Dresden  2  Dresden-Berlin 192 200  Dresden-Lajpcig 115 90  Dresden | Najkratka relacija: Dresden-Lajpcig (115 km) |
| 3  Dresden  2  Dresden-Berlin 192 200  Dresden-Lajpcig 115 90  Berlin  3  Berlin-Hanover 286 300  Berlin-Postdam 110 100  Berlin-Magdeburg 215 150  Minhen  4  Minhen-Stutgart 213 200  Minhen-Frankfurt 392 300  Minhen-Berlin 392 300  Minhen-Nuremberg 213 100  Minhen | Najkratka relacija: Minhen-Nuremberg (213 km) |
| 2  Dresden  2  Dresden-Berlin 192 200  Dresden-Lajpcig 115 90  Berlin  3  Berlin-Hanover 286 300  Berlin-Postdam 110 100  Berlin-Magdeburg 215 150  Berlin | Najkratka relacija: Berlin-Postdam (110 km) |

7. Во програмскиот јазик C да се креира структура Vozenje за опишување на адреналински возења во забавен парк. Структурата треба да содржи: **(5 поени)**

* Име на возењето ( низа од 100 знаци)
* Времетраење во минути (цел број)
* Цена (децимален број)
* Дали има студентски попуст (1-да, 0-не)

Да се креира структура ZabavenPark, што содржи: **(10 поени)**

* Име на забавниот парк (низа од 100 знаци)
* Локација (низа од 100 знаци)
* Низа од адреналински возења во паркот (низа од максимум 100 елементи од структурата Vozenje)
* Број на елементи во низата (цел број)

Да се креира функција print за печатење на информациите за сите возења во забавниот парк во формат: **(10 поени)**

[Ime na parkot] [Lokacija]

[Ime1] [Vremetraenje1] [Cena1]

[Ime2] [Vremetraenje2] [Cena2]

[Ime3] [Vremetraenje3] [Cena3]

...

Да се креира функција najdobar\_park, што прима низа од променливи од типот ZabavenPark и го печати името и локацијата на паркот кој е најефтин за посета од страна на студенти. Најефтин парк е оној што содржи најголем број возења со студентски попуст. Ако повеќе паркови ги делат овие карактеристики, да се испечати паркот со најголемото времетраење од сите возења.*\* (20 поени)*\*

Да се дополни функцијата main **(5 поени)**.

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| 2  Disneyland  Paris  2  BigThunderMountain  15  10  1  Shark  22  13  0  TivoliGardens  Copenhagen  3  Devil  16  11  1  FlyMountain  35  19  1  Nemo  8  7  0 | Disneyland Paris  BigThunderMountain 15 10.00  Shark 22 13.00  TivoliGardens Copenhagen  Devil 16 11.00  FlyMountain 35 19.00  Nemo 8 7.00  Najdobar park: TivoliGardens Copenhagen |

8. Во програмскиот јазик C да се креира структура Laptop за опишување на спецификациите на даден преносен компјутер. Структурата треба да содржи: **(5 поени)**

* Фирма која го произведува ( низа од 100 знаци)
* Големина на монитор во инчи (реален број)
* Дали е со touch или не (булова променлива)
* Цена (цел број)

Да се креира структура ITStore, што содржи: **(5 поени)**

* Име на продавницата (низа од 100 знаци)
* Локација (низа од 100 знаци)
* Низа од достапни лаптопи (низа од максимум 100 елементи од структурата Laptop)
* Број на елементи во низата (цел број)

Да се креира функција print за печатење на информациите за сите лаптопи во една продавница во формат: **(10 поени)**

[Ime na prodavnicata] [Lokacija]

[Marka1] [dim\_monitor1] [Cena1]

[Marka2] [dim\_monitor2] [Cena2]

[Marka3] [dim\_monitor3] [Cena3]

...

Да се креира функција najeftina\_ponuda, што прима низа од променливи од типот ITStore и го печати името и локацијата на онаа продавница која нуди најевтин преносен компјутер и неговата цена. Најефтин преносен компјутер е оној што има најниска цена и плус опција за touch. Ако има повеќе такви продавници, се печати прво најдената.*\* (15 поени)*\*

Да се дополни функцијата main **(10 поени)**.

Од тастатура се внесува бројот на продавници, па потоа за секоја продавница се внесуваат името и локацијата, а потоа бројот на компјутери, па за секој од компјутерите фирма која го произведува, големина на монитор, дали има touch и цена. Потоа се печатат сите внесени продавници и најевтината понуда.

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| 2  AMC  Skopje  3  dell  13.1  1  32500  asus  15.0  0  20200  hp  13.1  1  42000  Anhoch  Skopje  1  asus  13.3  1  32000 | AMC Skopje  dell 13.1 32500  asus 15 20200  hp 13.1 42000  Anhoch Skopje  asus 13.3 32000  Najeftina ponuda ima prodavnicata:  Anhoch Skopje  Najniskata cena iznesuva: 32000 |
| 1  AMC  Skopje  3  asus  15.0  1  20200  dell  13.3  1  32500  hp  13.3  1  42000 | AMC Skopje  asus 15 20200  dell 13.3 32500  hp 13.3 42000  Najeftina ponuda ima prodavnicata:  AMC Skopje  Najniskata cena iznesuva: 20200 |
| 1  AMC  Skopje  3  hp  13.3  1  42000  asus  15.0  0  20200  dell  13.3 1 32500 | AMC Skopje  hp 13.3 42000  asus 15 20200  dell 13.3 32500  Najeftina ponuda ima prodavnicata:  AMC Skopje  Najniskata cena iznesuva: 32500 |

9. Во програмскиот јазик C++ да се креира структура Laptop за опишување на спецификациите на даден преносен компјутер. Структурата треба да содржи: **(5 поени)**

* Фирма која го произведува ( низа од 100 знаци)
* Големина на монитор во инчи (реален број)
* Дали е со touch или не (булова променлива)
* Цена (цел број)

Да се креира структура ITStore, што содржи: **(5 поени)**

* Име на продавницата (низа од 100 знаци)
* Локација (низа од 100 знаци)
* Низа од достапни лаптопи (низа од максимум 100 елементи од структурата Laptop)
* Број на елементи во низата (цел број)

Да се креира функција print за печатење на информациите за сите лаптопи во една продавница во формат: **(10 поени)**

[Ime na prodavnicata] [Lokacija]

[Marka1] [dim\_monitor1] [Cena1]

[Marka2] [dim\_monitor2] [Cena2]

[Marka3] [dim\_monitor3] [Cena3]

...

Да се креира функција najeftina\_ponuda, што прима низа од променливи од типот ITStore и го печати името и локацијата на онаа продавница која нуди најевтин преносен компјутер и неговата цена. Најефтин преносен компјутер е оној што има најниска цена и плус опција за touch. Ако има повеќе такви продавници, се печати прво најдената.*\* (15 поени)*\*

Да се дополни функцијата main **(10 поени)**.

Од тастатура се внесува бројот на продавници, па потоа за секоја продавница се внесуваат името и локацијата, а потоа бројот на компјутери, па за секој од компјутерите фирма која го произведува, големина на монитор, дали има touch и цена. Потоа се печатат сите внесени продавници и најевтината понуда.

**ИСТИТЕ ТЕСТ ПРИМЕРИ ОД ПРЕТХОДНАТА ЗАДАЧА БИДЕЈЌИ СЕ ИСТИТЕ САМО ЕДНАТА ВО С++ ДРУГАТА ВО С!!!**

10. Да се креира структура Igrac во која се чуваат податоци за еден играч на компјутерската игра. За играчот се чуваат информации за корисничкото име на играчот (низа од 15 знаци), број на ниво до кој играчот е стигнат (цел број) и број на освоени поени (цел број) (10 поени).

Потоа да се креирa структура KompjuterskaIgra во која се чуваат податоци за името на играта (низа од 20 знаци), низа од играчи кои ја играат играта (најмногу 30) и бројот на играчи. (10 поени)

Треба да се направи функција со потпис (15 поени)

void najdobarIgrac(KompjuterskaIgra \*lista, int n)

Оваа функција треба да го испечати името на најдобриот играч на онаа игра која има најголема популарност (ја играат наjголем број на играчи). Најдобар играч е играчот кој има најголем број на освоени поени. Ако има повеќе играчи со ист максимален број на освоени поени, најдобар е играчот кој има достигнато најголемо ниво.

Печатењето е во форматот: "Najdobar igrac e igracot so korisnicko ime XXXXX koj ja igra igrata YYYYYY". (XXXXXX е корисничкото име на најдобриот играч,а YYYYY е името на играта која ја играат најголем број на играчи)

(Забелешка: секогаш ќе има точно еден најдобар играч)

Функционалност на задачата и дополнување на main функцијата (10 поени)

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| 2  Snake  2  user1 5 120  user2 4 120  Cars  3  koliubavi 5 130  fikjo 3 150  motor 4 140 | Najdobar igrac e igracot so korisnicko ime fikjo koj ja igra igrata Cars |
| 1  Snake  2  ana 5 120  ivana 4 120 | Najdobar igrac e igracot so korisnicko ime ana koj ja igra igrata Snake |
| 1  Snake  2  ana 5 120  ivana 4 200 | Najdobar igrac e igracot so korisnicko ime ivan koj ja igra igrata Snake |

11. Да се дефинира класа Ucesnik за учесник во маратон за кој ќе се чуваат информации за:

* име (динамички алоцирана низа од знаци)
* пол (bool)
* возраст (цел број) **(5 поени)**.

За потребите на оваа класа да се преоптоварат потребните конструктори и да се напише соодветен деструктор. **(10 поени)**

Дополнително за оваа класа да се преоптоварат:

* оператор за споредување > (споредува двајца учесници според возраста) **(5 поени)**
* операторот за проследување на излезен тек << кој ќе ги печати името, полот (машки/женски) и возраста, секој во посебен ред. **(5 поени)**

Да се дефинира класа Maraton за кој се чуваат:

* локација (низа од максимум 100 знаци)
* низа од Ucesnik објекти (динмички алоцирана низа од објекти од класата Ucesnik)
* број на учесници (цел број) **(5 поени)**.

За потребите на оваа класа да се преоптоварат потребните конструктори и да се напише соодветен деструктор. **(10 поени)** Исто така да се имплементираат следните методи:

* оператор += за додавање нов Ucesnik на маратонот Maraton. **(10 поени)**
* prosecnoVozrast() вреќа просечена возраст на учесниците на маратонот **(5 поени)**
* pecatiPomladi(Ucesnik &u) ги печати сите учесници помлади од учесникот проследен како аргумент на методот. **(5 поени)**

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| 5  Skopje  Elena 0 23  Mitko 1 41  Iskra 0 31  Jovan 1 28  Zoran 1 40 | Elena  zhenski  23  Iskra  zhenski  31  Jovan  mashki  28  32.6 |

12. Да се дефинира класа Пица за која ќе се чуваат податоци за:

* име на пицата (низа од 15 знаци)
* цена (цел број)
* состојки (динамички алоцирана низа од знаци)
* намалување на цената во проценти (цел број) - ако пицата не е на промоција намалувањето има вредност нула, во спротивно, вредност поголема од нула и не поголема од 100.

За потребите на оваа класа да се креираат потребните конструктори и да се напише соодветен деструктор. Дополнително за оваа класа да се дефинира функцијата:

* pecati() - функција во која ќе се печатат податоците за пицата во следниот формат: име - состојки, цена.
* istiSe(Pica p) - функција за споредба на две пици според состојките :

Да се дефинира класа Picerija во која се чуваат:

* име (низа од максимум 15 знаци)
* динмички алоцирана низа од објекти од класата Pica
* број на пици (цел број)

За потребите на оваа класа да се дефинираат потребните конструктори и да се напише соодветен деструктор. Да се имплементираат и следниве методи:

* dodadi(Pica P) - за додавање нова пица во пицеријата, но само ако не постои иста во низата (нема пица со исти состојки со пицата што треба да се додаде).
* void piciNaPromocija() - се печатат сите пици од пицеријата што се на промоција во формат : име - состојки, цена, цена со попуст.

Graphical user interface, text, application, email

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

13. Да се креира класата Zichara во која ќе се чуваат информации за:

* **место** (динамичко алоцирано поле од знаци)
* **цена на дневна карта** (цел број)

Да се креира класа PlaninarskiDom во која се чуваат информации за:

* **име** (статичко поле од најмногу 15 знаци)
* **цени по сезони** (поле од 2 цели броеви кои ја претставуваат цената за спиење за еден ден во планинарскиот дом во двете различни сезони. Секој планинарски дом има две сезони: летна и зимска. Летната сезона почнува на 1ви април и завршува до 1ви сепмтември. На пример цената за престој на 24ти Јули ќе биде сместена во полето со индекс 0 - каде е сместена цената за летната сезона)
* **класа на планинарскиот дом** (еден знак кој може да има вредност од A до F, каде A е најдобра класа)
* **дали има жичара** (bool-тип)
* **жичарата на планинарскиот дом** (покажувач кон објект од класата Zichara)

За дадените класи дефинирајте ги потребните: конструктори, деструктори, set и get методи. Оптоварете го соодветно операторот =. **(20 поени)**

Да се оптоварат:

* **операторот --** (префискен) за намалување на класата на планинарскиот дом (во класата PlaninarskiDom). Да се запази дека за планинарскиот дом најмала класа е F. **(5 поени)**
* **операторот <<** за печатење на објект одPlaninarskiDom. **(5 поени)**
* **операторот <=** за одредување дали планинарскиот дом има класа помала или еднаква со дадениот знак (споредба на планинарски дом со знак). **(5 поени)**

Формат на печатење на информации за еден планинарски дом:

* Pelister klasa:A so zichara **//ако има жичара**
* Korab klasa:B **//ако нема жичара**

Во класата PlaninarskiDom да се имплементира функција со име presmetajDnevenPrestoj со која се пресметува цената за дневен престој во планинарскиот дом за даден ден и месец во годината. Ако планинарскиот дом има жичара во цената за дневен престој влегува и дневната карта за жичарата. Функцијата враќа void (не враќа ништо). Првиот аргумент на функцијата е денот (1-31), а вториот е месецот (1-12) за кои се пресметува цената. Третиот аргумент е цената за дневен престој која ќе се пресмета во функцијата. Доколку денот или месецот се невалидни да се фрли соодветен исклучок со кој се справуваме во main-функцијата. Да се земе дека секој месец има 31 ден. **(15 поени)**

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| Pelister 21 31 E  1  Molika 20  3 3 | Informacii za PlaninarskiDomot:  Pelister klasa:F so Zichara  Planinarskiot dom za koj se vneseni informaciite ima klasa poniska ili ista so D  Cenata za 3.3 e 51 |
| Korab 48 42 A  0  12 4 | Informacii za PlaninarskiDomot:  Korab klasa:C  Cenata za 12.4 e 48 |
| Pelister 21 31 E  1  Molika 20  32 3 | Mesecot ili denot e greshno vnesen! |

14. Потребно е да се имплементираат класи File (датотека) и Folder (директориум) за работа со едноставен податочен систем.

Во класата File треба да се чуваат следниве податоци:

* Име на датотеката (динамички алоцирана низа од знаци)
* Екстензија на датотеката (енумерација со можни вредности: txt, pdf, exe)
* Име на сопственикот на датотеката (динамички алоцирана низа од знаци)
* Големина на датотеката (цел број на мегабајти)

Дополнително, во класата потребно е да се напишат и:

* Конструктор без параметри
* Конструктор со параметри
* Конструктор за копирање
* Деструктор
* Преоптоварување на операторот =
* Метод за печатење на информациите за една датотека (видете го излезот од тест примерите за структурата на печатење) - print()
* Метод за проверка на еднаквост помеѓу две датотеки со потпис bool equals(const File & that) кој ќе враќа true ако датотеките имаат исто име, екстензија и сопственик
* Метод за споредба на типот помеѓу две датотеки со потпис bool equalsType(const File & that) кој ќе враќа true ако датотеките се од ист тип, т.е. имаат исто име и екстензија

Во класата Folder треба да се чуваат следниве податоци:

* Име на директориумот (динамички алоцирана низа од знаци)
* Број на датотеки во него (на почеток директориумот е празен)
* Динамички алоцирана низа од датотеки, објекти од класата File

Дополнително, во класата потребно е да се напишат и:

* Конструктор со потпис Folder(const char\* name)
* Деструктор
* Метод за печатење на информациите за еден директориум (видете го излезот од тест примерите за структурата на печатење) - print()
* Метод за бришење на датотека од директориумот со потпис void remove(const File & file) кој ќе ја избрише првата датотека од директориумот која е еднаква според equals методот
* Метод за додавање на датотека во директориумот со потпис void add(const File & file) кој ќе додава датотеката во директориумот

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| 5  results  3  rezultati\_oop  FINKI  1  2  rezultati\_aok  FINKI  2  1  rezultati\_dm  FINKI  3  1  rezultati\_dm  FINKI  2  1 | ======= REMOVE FILE FROM FOLDER =======  Folder name: results  File name: rezultati\_oop.exe  File owner: FINKI  File size: 1  File name: rezultati\_aok.txt  File owner: FINKI  File size: 2 |
| 2  rezultati\_oop  FINKI  2  0  rezultati\_oop  FINKI  3  0  rezultati\_oop  FINKI  1  1 | ======= FILE EQUALS & EQUALS TYPE =======  File name: rezultati\_oop.pdf  File owner: FINKI  File size: 2  File name: rezultati\_oop.pdf  File owner: FINKI  File size: 3  File name: rezultati\_oop.txt  File owner: FINKI  File size: 1  FIRST EQUALS SECOND: TRUE  FIRST EQUALS THIRD: FALSE  FIRST EQUALS TYPE SECOND: TRUE  SECOND EQUALS TYPE THIRD: FALSE |
| 1  rezultati\_oop  FINKI  2  0 | ======= FILE CONSTRUCTORS AND = OPERATOR =======  ======= CREATED =======  File name: rezultati\_oop.pdf  File owner: FINKI  File size: 2  ======= COPIED =======  File name: rezultati\_oop.pdf  File owner: FINKI  File size: 2  ======= ASSIGNED =======  File name: rezultati\_oop.pdf  File owner: FINKI  File size: 2 |
| 3  results | ======= FOLDER CONSTRUCTOR =======  Folder name: results |
| 4  results  3  rezultati\_oop  FINKI  2  0  rezultati\_oop  Ministerot  2  0  rezultati\_oop  Ministerot  2  1 | ======= ADD FILE IN FOLDER =======  Folder name: results  File name: rezultati\_oop.pdf  File owner: FINKI  File size: 2  File name: rezultati\_oop.pdf  File owner: Ministerot  File size: 2  File name: rezultati\_oop.txt  File owner: Ministerot  File size: 2 |

15. Да се дефинира структура RabotnaNedela во која се чуваат бројот на работни часови за секој работен ден во неделата (5 дена) и број на неделата. Потоа да се дефинира структура Rabotnik за кој се чува име (низа од 50 знаци) и низа од работни недели (макс 4).

Да се имплементираат следните функции:

* int maxNedela(Rabotnik \*r) која за проследениот работник ќе го врати бројот на неделата со најмногу работни часови
* void table(Rabotnik \*r, int n) која за низата од работници ќе отпечати приказ во следниот формат:

**Пример**

Rab 1 2 3 4 Vkupno

Brianna 192 238 226 225 881

Ava 236 231 184 212 863

Matthew 236 235 219 243 933

Alyssa 240 254 217 228 939

Emma 247 205 204 242 898

каде во секој ред се печати името на работникот и вкупниот број на работни часови за секоја работна недела, како и вкупниот број на часови за сите недели. Во рамките на редот податоците се разделени со знак за табулатор \t.

|  |  |
| --- | --- |
| IPUT | OUTPUT |
| 10  Andrew  44 55 48 34 30  37 35 31 55 59  36 55 57 37 32  57 45 60 42 46  Mia  58 58 56 57 51  56 55 44 56 38  58 34 36 50 45  37 51 45 31 45  Natalie  50 59 42 33 43  44 30 48 38 59  53 34 53 52 31  40 45 44 41 59  John  37 43 39 59 49  51 60 49 60 33  40 38 58 39 46  42 33 31 42 60  Brianna  60 51 52 57 40  55 31 55 34 59  32 58 32 49 40  38 41 50 48 35  Samantha  40 35 45 55 38  39 31 56 55 40  55 46 38 36 32  46 31 56 47 57  Samantha  38 40 54 57 45  31 33 39 47 55  32 49 46 33 36  30 51 44 39 41  Ashley  32 39 49 31 38  45 54 41 45 55  41 59 47 35 30  45 43 40 37 53  Jacob  44 48 51 45 60  56 48 31 54 35  48 50 38 30 32  55 41 60 53 57  John  50 46 59 59 42  44 31 35 37 39  60 58 45 60 60  52 35 55 37 37 | TABLE  Rab\t1\t2\t3\t4\tVkupno  Andrew\t211\t217\t217\t250\t895  Mia\t280\t249\t223\t209\t961  Natalie\t227\t219\t223\t229\t898  John\t227\t253\t221\t208\t909  Brianna\t260\t234\t211\t212\t917  Samantha\t213\t221\t207\t237\t878  Samantha\t234\t205\t196\t205\t840  Ashley\t189\t240\t212\t218\t859  Jacob\t248\t224\t198\t266\t936  John\t256\t186\t283\t216\t941  MAX NEDELA NA RABOTNIK: Samantha  4 |
| 5  Brianna  35 33 42 42 40  31 57 52 54 44  56 32 51 52 35  43 39 34 54 55  Ava  44 47 44 50 51  30 46 59 36 60  39 32 35 31 47  44 38 41 46 43  Matthew  51 37 39 55 54  36 60 54 38 47  41 57 41 39 41  56 34 43 54 56  Alyssa  59 55 57 37 32  60 54 50 37 53  54 38 34 36 55  57 55 36 41 39  Emma  58 54 42 49 44  30 45 47 44 39  52 39 44 32 37  40 50 44 56 52 | TABLE  Rab\t1\t2\t3\t4\tVkupno  Brianna\t192\t238\t226\t225\t881  Ava\t236\t231\t184\t212\t863  Matthew\t236\t235\t219\t243\t933  Alyssa\t240\t254\t217\t228\t939  Emma\t247\t205\t204\t242\t898  MAX NEDELA NA RABOTNIK: Matthew  4 |

16. Да се дефинира структура RabotnaNedela во која се чуваат бројот на работни часови за секој работен ден во неделата (5 дена) и број на неделата. Потоа да се дефинира структура Rabotnik за кој се чува име (низа од 50 знаци) и низа од работни недели (макс 4).

Да се имплементираат следните функции:

* int maxNedela(Rabotnik \*r) која за проследениот работник ќе го врати бројот на неделата со најмногу работни часови
* void table(Rabotnik \*r, int n) која за низата од работници ќе отпечати приказ во следниот формат:

**Пример**

Rab 1 2 3 4 Vkupno

Brianna 192 238 226 225 881

Ava 236 231 184 212 863

Matthew 236 235 219 243 933

Alyssa 240 254 217 228 939

Emma 247 205 204 242 898

каде во секој ред се печати името на работникот и вкупниот број на работни часови за секоја работна недела, како и вкупниот број на часови за сите недели. Во рамките на редот податоците се разделени со знак за табулатор \t.

**ИСТИТЕ ТЕСТ ПРИМЕРИ ОД ПРЕТХОДНАТА ЗАДАЧА БИДЕЈЌИ СЕ ИСТИТЕ САМО ЕДНАТА ВО С++ ДРУГАТА ВО С!!!**

17. Фирмата **FINKI Cloud** има потреба од решение за управување со image датотеките од различните оперативни системи кои ги користат при сетирање на нови сервери. За таа цел потребно е да се дефинираат следниве класи:

OperativenSistem

Во класата OperativenSistem треба да се чуваат следниве податоци:

* име на оперативниот систем (динамички алоцирана низа од знаци)
* верзија (float)
* тип (енумерација со можни вредности LINUX, UNIX, WINDOWS)
* големина (во GB) (float)

Дополнително, во рамките на класата потребно е да се дефинира:

* конструктори со и без параметри
* copy constructor
* destructor
* преоптоварување на операторот =
* метод за печатење на информациите - void pecati() (видете го излезот од тест примерите за структурата)
* метод за проверка на **еднаквост** помеѓу два оперативни системи (два оперативни системи се еднакви ако имаат исто име, верзија, тип и големина) - bool ednakviSe(const OperativenSistem &os)
* метод за споредба на **верзии** помеѓу два оперативни система - int sporediVerzija(const OperativenSistem &os), кој ќе враќа 0 доколку верзиите се исти, -1 доколку верзијата на оперативниот систем од аргументот е поголема и 1 во преостанатиот случај.
* метод за проверка дали два оперативни системи се од иста **фамилија**. Два оперативни системи се од иста фамилија доколку имаат исто име и се од ист тип. bool istaFamilija(const OperativenSistem &sporedba)

Repozitorium

Во класата Repozitorium треба да се чуваат следниве податоци:

* име на репозиториумот (низа од 20 знака)
* динамички алоцирана низа од OperativenSistem
* број на оперативни системи кои моментално се наоѓаат во репозиториумот (int)

Дополнително, во рамките на класата потребно е да се дефинира:

* конструктор Repozitorium(const char \*ime)
* деструктор
* метод void pecatiOperativniSistemi() кој ќе ги печати сите оперативни системи кои се дел од репозиториумот
* метод void izbrishi(const OperativenSistem &operativenSistem) кој ќе го избрише оперативниот систем проследен како аргумент, доколку тој се наоѓа во репозиториумот. Да се користи методот ednakviSe за споредба на два оперативни система.
* метод за додавање на оперативен систем (void dodadi(const OperativenSistem &nov)) кој ќе го има следново однесување:
  + Доколку веќе постои оперативен систем од иста фамилија (да се користи методот за проверка на фамилија) кој е постар од проследениот оперативен систем, тогаш постоечкиот оперативен систем се заменува со новиот.
  + Во секој друг случај, проследениот оперативен систем се додава како дополнителен во репозиториумот.

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| FINKICloud  2  Ubuntu  17.04  1  2.25  Ubuntu  16.04  1  1.25  Ubuntu  14.04  1  1.25 | Repozitorium: FINKICloud  Ime: Ubuntu Verzija: 17.04 Tip: 1 Golemina:2.25GB  Ime: Ubuntu Verzija: 16.04 Tip: 1 Golemina:1.25GB  =====Brishenje na operativen sistem=====  Repozitorium: FINKICloud  Ime: Ubuntu Verzija: 17.04 Tip: 1 Golemina:2.25GB  Ime: Ubuntu Verzija: 16.04 Tip: 1 Golemina:1.25GB |
| FINKICloud  2  Ubuntu  17.04  1  2.25  Ubuntu  16.04  1  1.25  Ubuntu  16.04  1  1.25 | Repozitorium: FINKICloud  Ime: Ubuntu Verzija: 17.04 Tip: 1 Golemina:2.25GB  Ime: Ubuntu Verzija: 16.04 Tip: 1 Golemina:1.25GB  =====Brishenje na operativen sistem=====  Repozitorium: FINKICloud  Ime: Ubuntu Verzija: 17.04 Tip: 1 Golemina:2.25GB |
| FINKICloud  2  Ubuntu  16.04  1  1.25  Ubuntu  17.04  1  2.25  Ubuntu  12.04  1  2.25 | Repozitorium: FINKICloud  Ime: Ubuntu Verzija: 17.04 Tip: 1 Golemina:2.25GB  =====Brishenje na operativen sistem=====  Repozitorium: FINKICloud  Ime: Ubuntu Verzija: 17.04 Tip: 1 Golemina:2.25GB |
| FINKICloud  4  Ubuntu  16.04  1  1.25  Windows  10  3  3.5  Fedora  27  1  2.25  FreeBSD  11  2  1.75  Fedora  25  1  2.25 | Repozitorium: FINKICloud  Ime: Ubuntu Verzija: 16.04 Tip: 1 Golemina:1.25GB  Ime: Windows Verzija: 10 Tip: 3 Golemina:3.5GB  Ime: Fedora Verzija: 27 Tip: 1 Golemina:2.25GB  Ime: FreeBSD Verzija: 11 Tip: 2 Golemina:1.75GB  =====Brishenje na operativen sistem=====  Repozitorium: FINKICloud  Ime: Ubuntu Verzija: 16.04 Tip: 1 Golemina:1.25GB  Ime: Windows Verzija: 10 Tip: 3 Golemina:3.5GB  Ime: Fedora Verzija: 27 Tip: 1 Golemina:2.25GB  Ime: FreeBSD Verzija: 11 Tip: 2 Golemina:1.75GB |

18. Да се креира структура SkiLift во која се чуваат податоци за името на ски лифтот (низа од 15 знаци), максимален број на корисници што може да опслужи на едно возење, дали е пуштен во функција.

Потоа да се креирa структура SkiCenter во која се чуваат податоци за името на скијачкиот центар, држава во која се наоѓа (низи од 20 знаци), низа од ски лифтови (макс 20) што ги има и број на ски лифтови.

Треба да се направи функција со потпис

void najgolemKapacitet(SkiCenter \*sc, int n)

која што ќе го отпечати ски центарот што има најголем капацитет за опслужување скијачи (вкупниот број на скијачи кои може да се опслужуваат во еден момент вкупно на сите ски лифтови во центарот). Доколку два ски центри имаат ист капацитет, тогаш се печати оној кој има поголем број на ски лифтови. Притоа треба да се испечатат во посебен ред името, државата и капацитетот на ски центарот. Кога се пресметува капацитет на еден ски центар во предвид се земаат само ски лифтовите кои се поставени дека се во функција.

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| 1  Mavrovo  Makedonija  6  Ednosed  30  1  Dvosed  60  1  Ciciban  10  1  Galicki  20  1  Belicki  36  0  Plavi  40  1 | Mavrovo  Makedonija  160 |

19. Да се дефинира класа IceCream во која се чуваат податоци за:

* име (динамичка низа од знаци),
* состав (низа од знаци, најмногу 100)
* цена (децимален број),
* попуст (цел број, изразен во проценти) иницијална вредност 0.

За потребите на класата да се имплементираат:

* соодветни конструктори и деструктор (да се погледнат повиците во функцијата main)
* оператор << за печатење на податоците за сладоледот во следниов формат:

име: состав цена [цена на попуст]

делот „цена на попуст“ се печати **само ако** за сладоледот се нуди попуст (вредноста на попустот е поголема од нула)

* оператор ++ во префикс нотација за зголемување на попустот за 5
* оператор + - за собирање на објект од класата IceCream со низа од знаци. Собирањето се реализира на начин што името на новиот сладолед се добива со спојување на името на сладоледот и низата од знаци одделени со „ + “. Цената на новиот сладолед се зголемува за 10, а попустот и составот на сладоледот остануваат исти.
* метод void setDiscount(int discount) со кој може да се смени вредноста на попустот. Дозволените вредности на попустот се во интервалот [0,100]
* метод void setName(char\* name) со кој се менува името на сладоледот.

Да се креира класа IceCreamShop во која се чува:

* име на продавницата (низа од знаци, најмногу 50)
* динамички резервирано поле од објекти од класата IceCream
* број на сладоледи во полето (цел број).

За оваа класа да се обезбедат:

* соодветни конструктори и деструктор (погледнете ги повиците во main функцијата)
* оператор += за додавање нов објект од класата IceCream во полето
* оператор << за печатење на листа од сите сладоледи во продавницата. Прво се печати името на продавницата, а потоа се печатат податоците за сите сладоледи одделени со нов ред.

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| 3  FINKI Ice Cream  3  Strawberry Ice Cream in Lemon Bowls  Strawberry ice cream, Lemons  100  Caramel Apple Delight  Caramel Delight ice cream, Apples  120  Dreyer’s Ice Cream Blueberry Pie  French Vanilla ice cream, Graham cracker pie crust, blueberries  60 | ====== TESTING IceCreamShop ======  CONSTRUCTOR  OPERATOR +=  FINKI Ice Cream  Strawberry Ice Cream in Lemon Bowls: Strawberry ice cream, Lemons 100  Caramel Apple Delight: Caramel Delight ice cream, Apples 120  Dreyer’s Ice Cream Blueberry Pie: French Vanilla ice cream, Graham cracker pie crust, blueberries 60 |
| 2  Caramel Apple Delight  Caramel Delight ice cream, Apples  60  Caramel Apple Delight Ice Cream in Apples  Special Ice Cream  -40 | ====== TESTING IceCream CONSTRUCTORS ======  CONSTRUCTOR  Caramel Apple Delight: Caramel Delight ice cream, Apples 60  COPY CONSTRUCTOR  Caramel Apple Delight: Caramel Delight ice cream, Apples 60  Caramel Apple Delight Ice Cream in Apples: Caramel Delight ice cream, Apples 60  OPERATOR =  Caramel Apple Delight Ice Cream in Apples: Caramel Delight ice cream, Apples 60  Special Ice Cream: Caramel Delight ice cream, Apples 60 |
| 1  Strawberry Ice Cream in Lemon Bowls  Strawberry ice cream, Lemons  80  10  Caramel Apple Delight  Caramel Delight ice cream, Apples  60  20 | ====== TESTING IceCream CLASS ======  CONSTRUCTOR  OPERATOR <<  Strawberry Ice Cream in Lemon Bowls: Strawberry ice cream, Lemons 80 (72)  Caramel Apple Delight: Caramel Delight ice cream, Apples 60 (48)  OPERATOR ++  Strawberry Ice Cream in Lemon Bowls: Strawberry ice cream, Lemons 80 (68)  OPERATOR +  Caramel Apple Delight + chocolate: Caramel Delight ice cream, Apples 70 (56) |
| 4  FINKI Ice Cream  3  Strawberry Ice Cream in Lemon Bowls  Strawberry ice cream, Lemons  100  Caramel Apple Delight  Caramel Delight ice cream, Apples  120  Dreyer’s Ice Cream Blueberry Pie  French Vanilla ice cream, Graham cracker pie crust, blueberries  80 | ====== TESTING IceCreamShop CONSTRUCTORS ======  FINKI Ice Cream  Strawberry Ice Cream in Lemon Bowls: Strawberry ice cream, Lemons 100  Caramel Apple Delight: Caramel Delight ice cream, Apples 120  Dreyer’s Ice Cream Blueberry Pie: French Vanilla ice cream, Graham cracker pie crust, blueberries 80  FINKI fruits: strawberry ice cream, raspberry ice cream, blueberry ice cream 60  FINKI Ice Cream  Strawberry Ice Cream in Lemon Bowls: Strawberry ice cream, Lemons 100  Caramel Apple Delight: Caramel Delight ice cream, Apples 120  Dreyer’s Ice Cream Blueberry Pie: French Vanilla ice cream, Graham cracker pie crust, blueberries 80 |

20. Во програмскиот јазик C да се креира структура Pacient и истата треба да содржи: (5 поени)

* Име и презиме на пациентот ( низа од макс. 100 знаци)
* Дали е здраствено осигуран (1-да, 0-не)
* Број на прегледи во текот на последниот месец (цел број)

Да се креира структура MaticenDoktor, која во себе содржи: (5 поени)

* Име на докторот (низа од 100 знаци)
* Број на пациенти (цел број)
* Листа од пациенти (низа од максимум 200 елементи од структурата Pacient)
* Цена на преглед (децимален број)

Да се креираат метода najuspesen\_doktor, која прима низа од променливи од типот MaticenDoktor и нивниот број; и го печати името, заработената сума и бројот на прегледи на оној доктор кој заработил најголема сума од пациенти кои ги прегледувал приватно (не се здраствено осигурани) (10 поени). Доколку два или повеќе доктори имаат иста сума, тогаш се печати оној доктор кој има направено најголем број на прегледи вкупно во текот на последниот месец. (20 поени)

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| 3  Д-р.Јованка\_Јовановска  3  300.00  Јованова\_Ивана 1 2  Боцевска\_Ивана 0 3  Илиевска\_Ангела 1 0  Д-р.Петко\_Петковски  1  50.00  Ивановска\_Цветанка 0 18  Д-р.Трајче\_Трајчев  3  300.00  Крстевски\_Филип 1 0  Цветановски\_Иван 1 10  Ивановска\_Цветанка 0 3 | Д-р.Петко\_Петковски 900.00 18 |
| 2  Д-р.Јованка\_Јовановска  3  300.00  Јованова\_Ивана 1 2  Боцевска\_Ивана 0 3  Илиевска\_Ангела 1 0  Д-р.Трајче\_Трајчев  3  300.00  Крстевски\_Филип 1 0  Цветановски\_Иван 1 10  Ивановска\_Цветанка 0 3 | Д-р.Трајче\_Трајчев 900.00 13 |
| 2  Д-р.Јованка\_Јовановска  3  300.00  Јованова\_Ивана 1 2  Боцевска\_Ивана 0 3  Илиевска\_Ангела 1 0  Д-р.Трајче\_Трајчев  2  650.00  Крстевски\_Филип 1 0  Цветановски\_Иван 1 10 | Д-р.Јованка\_Јовановска 900.00 5 |

21. Да се креира структура Tanc во која се чуваат податоци за името на танцот и земја на потекло (низи од 15 знаци) .

Потоа да се креирa структура Tancer во која се чуваат податоци за име и презиме (низи од максимум 20 знаци), низа од 5 танци (структура Tanc) кои дадениот танчер може да ги игра. Дополнително, за оваа структура да се обезбеди функција со потпис :

void tancuvanje(Tancer \*t, int n, char \*zemja)

што ќе ги испечати во посебен ред името и презимето на танчерите што знаат да играат барем по еден танц од земјата која е проследена како аргумент на функцијата, како и името на танцот во формат: **Име Презиме, Име на танц**

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| 4  Aleks  Aleksov  samba  brazil  salsa  kuba  merengue  dominikana  Marko  Markov  tango  argentina  flamenco  spanija  samba  brazil  Nikola  Nikolov  chacha  kuba  samba  brazil  vienski  avstrija  Marta  Martovska  samba  brazil  samba1  brazil  bachata  dominikana  brazil | Aleks Aleksov, samba  Marko Markov, samba  Nikola Nikolov, samba  Marta Martovska, samba |
| 2  Aleks  Aleksov  samba  brazil  salsa  kuba  merengue  dominikana  Marko  Markov  tango  argentina  flamenco  spanija  samba  brazil  kuba | Aleks Aleksov, salsa |

22. Да се креира структура Proizvod во која се чуваат податоци за кодот напрозиводот (низа од 20 знаци), единична цена на производ (цел број), бројот на достапни производи кои ги има во еден магацин (цел број) .

Потоа да се креирa структура Narachka во која се чуваат податоци за името лицето кое ја прави нарачката (низи од 15 знаци), низа од Proizvodi (максимум 10) кои ги има во нарачката, низа од бројот на нарачани производи за секоја производ посебно (максимум10) и бројот на производи во нарачката .

Треба да се направи функција со потпис

void pecatiFaktura(Narachka n)

која што ќе ги отпечати една фактура за една нарачка (n) .

Во фактурата во првиот ред се печати: **Faktura za XXXXX**, каде XXXXX е името на лицето кое ја прави нарачката.

Ако сите производи може да се нарачаат печатат нарачаните производи, подредени по азбучен ред според кодот на производот. Во следните редови, во секој ред посебно се печати кодот на производот, единична цена на производот, број на нарачани производи и вкупната цена на производот (сите се одделени со едно празно место) . Исто така бројот на достапни производи од тој производ да се ажурира (да се намали за бројот на нарачани производи од тој производ). Во последниот ред да се отпечати: **Vkupnata suma na fakturata e XXX**, каде XXX е вкупната сума за цените за сите нарачани производи.

Ако постои производ кој не може да биде нарачан затоа што во магацинот не се достапни нарачаниот број производи тогаш не може да се изготви фактура за нарачката. На екранот да се отпечати: **Fakturata ne moze da se izgotvi**.

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| Stefan 2  proizvod1 10 1  proizvod2 20 3  3 2 | Faktura za Stefan  Fakturata ne moze da se izgotvi |
| Marko 2  proizvod1 10 4  proizvod2 20 3  3 2 | Faktura za Marko  proizvod1 10 3 30  proizvod2 20 2 40  Vkupnata suma na fakturata e 70 |
| Stefan 3  fanta 53 100  vino 130 40  voda 22 150  50 10 30 | Faktura za Stefan  fanta 53 50 2650  vino 130 10 1300  voda 22 30 660  Vkupnata suma na fakturata e 4610 |
| Stefan 3  voda 22 150  fanta 53 100  vino 130 40  50 10 30 | Faktura za Stefan  fanta 53 50 2650  vino 130 10 1300  voda 22 30 660  Vkupnata suma na fakturata e 4610 |