СОФИЙСКИ УНИВЕРСИТЕТ
“СВ. КЛИМЕНТ ОХРИДСКИ”

ФАКУЛТЕТ ПО МАТЕМАТИКА
И ИНФОРМАТИКА

ДЪРЖАВЕН ИЗПИТ

ЗА ПОЛУЧАВАНЕ НА ОКС “БАКАЛАВЪР ПО ИНФОРМАЦИОННИ СИСТЕМИ”

ЧАСТ I (ПРАКТИЧЕСКИ ЗАДАЧИ)

Драги абсолвенти:

- Попълнете факултетния си номер в горния десен ъгъл на всички листове.
- Пишете само на предоставените листове, без да ги разкопчавате.
- Решението на една задача трябва да бъде на същия лист, на който е и нейното условие (т.е. може да пишете отпред и отзад на листа със задачата, но не и на лист на друга задача).
- Ако имате нужда от допълнителен лист, можете да поискате от квесторите.
- На един лист не може да има едновременно и чернова, и белова.
- Черновите трябва да се маркират, като най-отгоре на листа напишете “ЧЕРНОВА”.
- Ако решението на една задача не се побира на нейния лист, трябва да поискате нов бял лист от квесторите. Той трябва да се защипе с телбод към листа със задачата.
- Всеки от допълнителните листове (белова или чернова) трябва да се надпише най-отгоре с вашия факултетен номер.
- Черновите също се предават и се защипват в края на работата.
- Времето за работа по изпита е 3 часа.

Изпитната комисия ви пожелава успешна работа!

Задача 1. а) Имената на служители в дадена компания и на техните преки ръководители можем да представим с двумерен масив `const char* leaders[][2]`.

Например:

Служител	Ръководител
Иван Иванов	Мария Иванова
Мария Иванова	Иван Драганов
Иван Драганов	Стоян Петров

Казваме, че служителят *A* е ръководител на служителя *B*, ако *A* е пряк ръководител на *B* или е пряк ръководител на ръководител на *B*. Да се дефинира рекурсивна функция:

```
bool is_subordinate (const char* employee,
                     const char* manager,
                     const char* leaders[][2],
                     size_t n),
```

която проверява дали служителят с име `employee` е подчинен на служителя с име `manager` в компанията, описана с масива `leaders` с `n` реда.

б) Да се дефинира функция

`const char* the_big_boss(const char* leaders[][2], size_t n)`,

намираща името на служителя, който се намира най-високо в йерархията на компанията, описана с масива `leaders` с `n` реда.

Приемаме, че йерархията от служителите, описана в `leaders` е коректна, т.е. всеки служител има по точно един пряк ръководител, с изключение на точно един служител, който няма пряк ръководител, и няма двама служители такива, че всеки от тях е ръководител на другия.

Да се демонстрира извикването на функциите с кратка програма.

Задача 2. а) Да се проектират и разработят класове `Point2D` и `PointSet2D`. `Point2D` да описва точка в евклидовата равнина. Координатите на точката да се представят с числа с плаваща запетая (от тип `double`). `PointSet2D` да описва множество от точки в равнината и да предоставя следните възможности:

1. Добавяне на произволен брой точки към множеството. Да не се допускат повторения на елементи на множеството. Добавянето на точки да може да се извършва чрез операциите `+` и `+=`.
2. Операции `+` и `+=` за обединение на две множества от точки.
3. Възможност за обхождане на точките от множеството.

б) Да се дефинира булева функция `within`, която проверява дали всички точки на дадено множество лежат в кръг с даден център и радиус.

При проектирането на класовете да се спазят следните изисквания:

- Да се спази принципът на капсулацията.
- Да се реализират подходящи конструктори, деструктори, операции за присвояване и др., които са подходящи за класовете.
- Операциите `+` и `+=` да са с коректно подбрана мутираща или немутираща (константна) форма, да имат подходящ тип на резултата и да връщат подходящи резултати.

Забележка: позволено е използването на класовете контейнери от стандартната библиотека `STL`.

Задача 3. *Задачата да се реши на един от езиците Scheme или Haskell.*

Музикално произведение се описва с наредена тройка от име на автор, заглавие на произведението и дължина в секунди. Даден е непразен списък от музикални произведения `pl`, в който **те са подредени в нарастващ ред по дължина**. *Препоръчваща функция* наричаме функция, която приема наредена тройка, описваща текущо възпроизвеждащо се музикално произведение и връща заглавие на произведение, което се препоръчва да бъде следващото възпроизведено. Компанията *λify* е патентовала алгоритъм за препоръчваща функция, която за текущо възпроизвеждащо се произведение `p` с първи компонент (автор) `a` работи по следния начин за списък `pl`:

1. Препоръчва следващото по дължина произведение на `a` в `pl`, ако има такова.
2. В противен случай, препоръчва произведение с максимална дължина в `pl` из произведенията на всички автори, за които средната дължина на произведенията в `pl` е по-малка от средната дължина на произведенията на `a` в `pl`, ако има такива.
3. В противен случай, препоръчва следващото по дължина произведение в `pl` след `p`, а ако такова няма, препоръчва отново `p`.

Да се попълнят празните полета по-долу така, че функцията `recommender pl` да връща препоръчваща функция, реализираща алгоритъма на *λify*, описан по-горе, върху списък `pl`. Функцията `avgDuration` да връща средната дължина на произведенията на `author` в `pl`. Списъкът `option1` да съдържа произведенията на `a` с по-голяма дължина от `p`. Списъкът `option2` да съдържа произведенията на всички автори със средна дължина на произведенията по-малка от средната дължина на произведенията на `a`.

Упътване: *могат да се използват наготово функциите `append`, `apply`, `filter`, `foldr`, `fromIntegral`, `length`, `list`, `last`, `map`, `reverse`, `sum`, както и всички функции в R5RS за Scheme и в Prelude за Haskell.*

Scheme

```
(define (recommender pl)
  ____________________________
    (define (avgDuration artist) ____________________________________________)
    (define option1 ____________________________________________________________)
    (define option2 ____________________________________________________________)
    (if (not (null? option1)) ___________________ option1
        (if (not (null? option2)) ___________________ option2
            ____________________________________________________________)))
```

Пример:

```
(define rf (recommender '(("Mozart" "The Marriage of Figaro Overture" 270) ("Gershwin"
  "Summertime" 300) ("Queen" "Bohemian Rhapsody" 355) ("Gershwin" "Rhapsody in Blue" 1100))))
(rf '("Mozart" "The Marriage of Figaro Overture" 270)) ⟶ "Summertime"
(rf '("Gershwin" "Summertime" 300)) ⟶ "Rhapsody in Blue"
(rf '("Gershwin" "Rhapsody in Blue" 1100)) ⟶ "Bohemian Rhapsody"
```

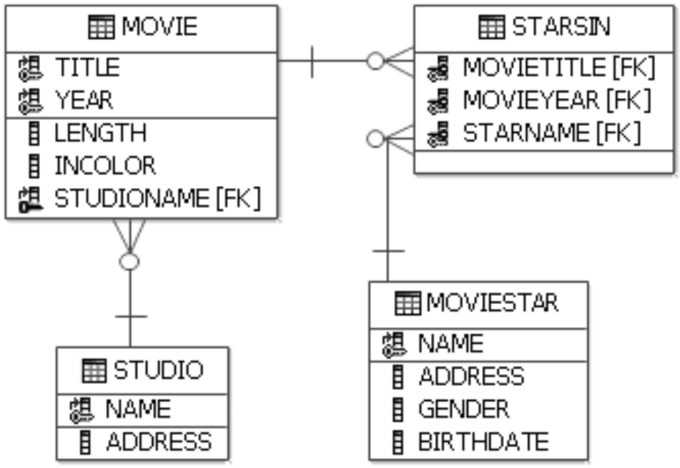
Haskell

```
recommender pl = ____________________________
  let avgDuration artist = ____________________________________________
      option1 = ____________________________________________________________
      option2 = ____________________________________________________________
  in if not (null option1) then (___________________ option1  )
     else if not (null option2) then (___________________ option2  )
          else ____________________________________________________________
```

Пример:

```
rf = recommender [("Mozart","The Marriage of Figaro Overture",270), ("Gershwin","Summertime",
       300), ("Queen","Bohemian Rhapsody",355), ("Gershwin","Rhapsody in Blue",1100)]
rf ("Mozart","The Marriage of Figaro Overture",270) ⟶ "Summertime"
rf ("Gershwin", "Summertime", 300) ⟶ "Rhapsody in Blue"
rf ("Gershwin", "Rhapsody in Blue", 1100) ⟶ "Bohemian Rhapsody"
```

Задача 4. Дадена е базата от данни `Movies`, в която се съхранява информация за филми, филмови студия, които ги произвеждат, както и актьорите, които участват в тях.



Таблицата `Studio` съдържа информация за филмови студиа:

- `name` — име, първичен ключ
- `address` — адрес;

Таблицата `Movie` съдържа информация за филми. Атрибутите `title` и `year` заедно формират първичния ключ.

- `title` — заглавие
- `year` — година, в която е заснет филмът
- `length` — дължина в минути
- `incolor` — `'Y'` за цветен филм и `'N'` за чернобял
- `studioname` — име на студио, външен ключ към `Studio.name`;

Таблицата `MovieStar` съдържа информация за филмови звезди:

- `name` — име, първичен ключ
- `address` — адрес
- `gender` — пол, `'M'` за мъж (актьор) и `'F'` за жена (актриса)
- `birthdate` — рождена дата.

Таблицата `StarsIn` съдържа информация за участието на филмовите звезди във филмите. Трите атрибута заедно формират първичния ключ. Атрибутите `movietitle` и `movieyear` образуват външен ключ към `Movie`.

- `movietitle` — заглавие на филма
- `movieyear` — година на заснемане на филма
- `starname` — име на филмовата звезда, външен ключ към `MovieStar.name`.

Зад 1. Да се напише заявка, която извежда имената и адресите на всички студиа, които имат поне един цветен и поне един черно-бял филм. Резултатът да се сортира възходящо по адрес.

Зад 2. Да се напише заявка, която за всяко студио с най-много три филма извежда:

- името му;
- адреса;
- средната дължина на филмите на това студио.

Студиа без филми също да се изведат (за средна дължина да се извежда `null` или 0).

Задача 5. Информационна система съхранява информация за онлайн пицария. В системата се съхраняват клиентите на пицарията – телефонен номер на клиент (уникален за всеки клиент) – низ до 12 символа, име на клиент – низ до 50 символа, адрес – низ до 150 символа. Онлайн пицарията предлага пици и напитки. Пиците се характеризират с идентификационен номер – низ от точно 5 символа (уникално за всяка пица), име – низ до 50 символа, цена – реално число и тегло в грамове – цяло положително число. Всяка пица се състои от основни продукти: основа, доматен сос, моцарела, гъби, шунка и т.н. и добавки, които клиентът може допълнително да добави към пицата – например допълнително моцарела, домат и т.н. Продуктите за пицата имат име – низ до 30 символа, грамаж – цяло положително число и цена. Напитките се характеризират с идентификационен номер – низ от точно 5 символа (уникален за всяка напитка), име – низ до 30 символа, цена – реално число, количество в литри – реално положително число. Клиентът прави поръчка за напитка или пица. Една поръчка може да включва много пици и напитки. Поръчката се прави точно от един клиент и един клиент може да направи много поръчки в един и същи ден.

Задание:

а) Да се направи E/R модел на БД, която съхранява гореописаната информация. Да се начертае E/R диаграма на модела.

б) Да се преобразува E/R диаграмата към релационни схеми. Да се премахнат излишествата, където това е възможно.

в) Да се напише DDL код, съответстващ на релационните схеми. Да се реализират всички описани ограничения.

Задача 6. В контекста на информационната система за онлайн пицария, представена в Задача 5 (за E/R моделиране), разглеждаме процесите по регистрация, управление на менюто, подаване и изпълнение на поръчка. Клиентът може да поръча пица от менюто или да създаде и запази пица по свой вкус, като избере от предлаганите в менюто съставки. Има възможност да избира за поръчка пица или напитка от минали поръчки, вместо от менюто, ако те са включени в актуалната версия на менюто. В поръчката може да се укаже адрес за доставка, който е различен от адреса на клиента, посочен при регистрацията му. Системата осигурява възможност на клиента да проследи състоянието на поръчката. Поръчката може да се заплати онлайн или при доставка.

Ако при изпълнението на подусловие а) и/ или б) са направени допускания във връзка с представения сценарий, те трябва да бъдат описани явно.

Задание:

а) Да се подготви диаграма на потребителските случаи в термините на UML стандарта. Да се добави кратко текстово описание на диаграмата.

б) Да се опише в пълен формат потребителски случай за подаване на поръчка към пицарията. В описанието да се включат минимум: актьори; предусловия; следусловия (резултати); основен сценарий; отношения с други потребителски случаи (ако е приложимо); един алтернативен и един негативен сценарий; какви проверки за коректност се правят при обработката, какви номенклатури се използват (ако е приложимо); специфични за потребителския случай допълнителни изисквания.

Задача 7. На всеки опит хвърляме три пъти последователно зар. Дефинираме събитие

A = {точките при някое хвърляне са равни на сумата от точките на другите две хвърляния}.

а) Да се определи вероятността на A при извършване на един опит.

б) Извършваме опити докато събитието A се изпълни. Нека X е броят на хвърлянията на зар, които сме направили при това. Да се намери математическото очакване $\mathrm{E}X$ и дисперсията $\mathrm{D}X$.

в) Колко опита трябва да бъдат направени, така че да е по-вероятно събитието A да се сбъдне поне веднъж, отколкото да не се сбъдне нито веднъж?

Чернова