**СОФИЙСКИ УНИВЕРСИТЕТ „СВ. КЛИМЕНТ ОХРИДСКИ“**

**ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА**

# ДЪРЖАВЕН ИЗПИТ
## ЗА ПОЛУЧАВАНЕ НА ОКС “БАКАЛАВЪР ПО ИНФОРМАЦИОННИ СИСТЕМИ”

### ЧАСТ I (ПРАКТИЧЕСКИ ЗАДАЧИ)
### 13.07.2018 г.

| Моля, не пишете в тази таблица! | | | |
|---|---|---|---|
| Зад. 1 | | Зад. 5 | |
| Зад. 2 | | Зад. 6 | |
| Зад. 3 | | Зад. 7 | |
| Зад. 4 | | Зад. 8 | |
| Крайна оценка: | | | |

Драги абсолвенти:

- Попълнете факултетния си номер в горния десен ъгъл на всички листа;
- Пишете само на предоставените листове без да ги разкопчавате;
- Ако имате нужда от допълнителен лист, можете да поискате от квесторите;
- Допълнителните листа трябва да се номерират, като номерата продължават тези от настоящия комплект;
- Всеки от допълнителните листа трябва да се надпише най-отгоре с вашия факултетен номер;
- **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 решението на задачата не се побира в един лист, трябва да поискате нов бял лист от квесторите. В такъв случай отново трябва да започнете своето решение на листа с условието на задачата и в края му да напишете „Продължава на лист № X”, където X е номерът на допълнителния лист, на който е вашето решение;
- Черновите трябва да бъдат маркирани, като най-отгоре на листа напишете „ЧЕРНОВА“;
- На един лист не може да има едновременно и чернова и белова;
- Времето за работа по изпита е 3 часа.

*Изпитната комисия ви пожелава успешна работа!*

Задача 1. *Задачата да се реши на C/C++.*

Да се дефинира функция **`sortLex`**, която получава като аргументи положителното число **`n`** и масив **`a`**, съдържащ **`n`** на брой цели неотрицателни числа, и ги сортира във възходящ ред относно лексикографската наредба (например, 123 е преди 9 по лексикографската наредба). Да се напише кратка програма, в която да се демонстрира използването на функцията **`sortLex`**.
За реализацията на функцията **`sortLex`** не е позволено използването на стандартни библиотечни функции.

**Пример:**
При подаден масив `{13,14,7,2018,9,0}`, след изпълнение на функцията **`sortLex`** масивът ще бъде подреден по следния начин: `{0,13,14,2018,7,9}`.

Задача 2. *Задачата да се реши на C++. Позволено е използване на функции и класове от стандартната библиотека.*

Да се реализира клас полином, коефициентите на който да бъдат от шаблонен тип **T**. За класа да се реализира конструктор, който приема масив от елементи от тип **T**, представящи коефициентите на полинома в намаляващ ред на степенните показатели, и цяло положително число, представящо степента на полинома. Предефинирайте поне следните операторни функции:

- двуместни оператори **+** и **−** с аргументи два полинома;
- двуместен оператор ***** с аргумент полином и елемент от тип **T**;
- оператор за индексиране, който връща коефициента пред съответната степен (при некоректна степен да се връща произволна стойност);
- оператор **()** с единствен аргумент **x** от тип **T**, който пресмята стойността на полинома в точката **x**.

За двуместните оператори да се реализират и съответните им варианти с присвояване **+=**, **−=** и ***=**, където това има смисъл.

Задача 3. *Задачата да се реши на един от езиците Scheme или Haskell. По-долу оградете името на езика, който сте избрали за вашето решение.*

Дадени са непразен списък от едноместни числови функции **`fl`** и непразен списък от числа **`xl`**. Казваме, че числото **`x`** е "неподвижна точка" на функцията **`f`**, ако **`f(x) = x`**. Да се попълнят по подходящ начин празните полета по-долу така, че за всички функции от **`fl`**, които имат неподвижна точка сред числата в **`xl`**, функцията **`sumMinFix`** да намира сумата на най-малките им такива неподвижни точки. Ако никоя функция от **`fl`** няма неподвижна точка сред числата в **`xl`**, функцията **`sumMinFix`** да връща числото 0. Помощната функция **`addDefault`** служи да осигури, че ако подаденият ѝ списък е празен, то в него се добавя една стойност по подразбиране.
**Упътване:** можете да използвате наготово функциите `apply`, `filter`, `foldr`, `map`, `min`, `minimum`, както и всички стандартни функции в R$^5$RS за Scheme и Prelude за Haskell.

## Scheme

```
(define (addDefault val l)
  (if (null? l) (list val) l))

(define (sumMinFix fl xl)
  (_________________
    (_________________
      (lambda (f)
        (apply _________________
          (addDefault _________________
            (_________________
              (lambda (x) _____________________________ ) xl)))) fl)))
```

**Пример:**

```
(sumMinFix (list (lambda (x) (/ 1 x)) exp (lambda (x) (- (* 2 x) 3))))
           '(-2 -1 1 3)) → 2  (= -1 + 3)
```

## Haskell

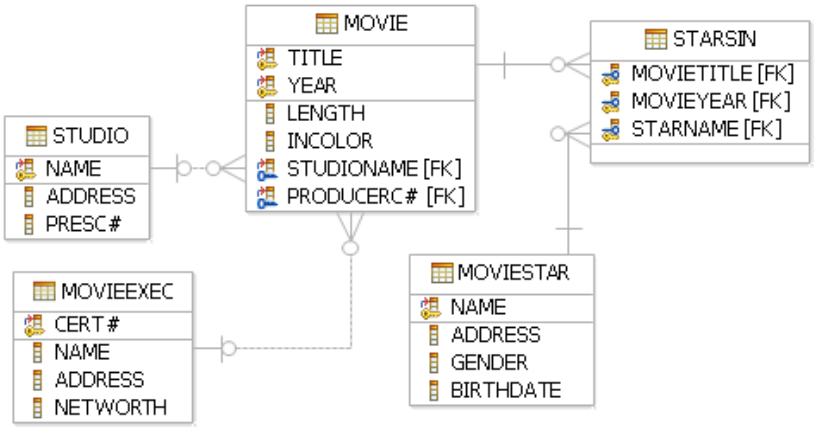
```
addDefault val [] = [val]
addDefault val l  = l

sumMinFix fl xl =

  _________________
     (_________________
        (\f -> _________________
                (addDefault _________________
                   [ _________________ | x <- xl, _____________________________ ])) fl)
```

**Пример:**

```
sumMinFix [ (1/), exp, \x -> 2*x - 3 ] [-2, -1, 1, 3] → 2   (= -1 + 3)
```

Задача 4. Дадена е базата от данни `Movies`, в която се съхранява информация за филми, филмови студиа, които ги произвеждат, продуцентите на филмите, както и актьорите, които участват в тях. 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`;
- `producerc#` — номер на сертификат на продуцента, външен ключ към `MovieExec.cert#`.




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`.

Таблицата `Studio` съдържа информация за филмови студиа:

- `name` — име, първичен ключ;
- `address` — адрес;
- `presc#` — номер на сертификат на президента на студиото.

Таблицата `MovieExec` съдържа информация за продуцентите на филми.

- `cert#` —номер на сертификат, първичен ключ;
- `name` — име;
- `address` — адрес;
- `networth` — нетни активи;

Забележка за всички таблици: Всички атрибути, които не участват във формирането на първичен ключ, могат да приемат стойност **`NULL`**.

1. Да се напише заявка, която да изведе име на студио, годината на първия филм за това студио, годината на последния филм за това студио и броя на всички филми за това студио, само за тези студиа, чиито имена започват с буквата ‘M’.
2. Да се напише заявка, която да изведе името на актрисата, участвала в най-много филми, и броя на филмите, в които е участвала.

Задача 5. Информационна система съхранява информация за кандидат-студентска кампания (КСК) на университет. Съхранява се информация за кандидат-студенти с данни за техния входящ номер — уникален за всеки кандидат-студент, ЕГН, име на кандидат-студента. Съхранява се също информация и за изпити с данни за код на изпита — уникален за всеки изпит, име на изпита — уникално за всеки изпит, дата и час на провеждане на изпита. Един кандидат-студент може да заяви участие за много изпити и за един изпит може да са заявили участие много кандидат-студенти. Във всяко заявление на кандидат-студента трябва да се пази и информация за дата на подаване на заявлението и сесията, на която кандидат студента се явява (цяло положително число). В информационната система се съхраняват и данни за факултети с код на факултета — уникален за всеки факултет, име на факултет — уникално за всеки факултет и адрес на факултета. Един факултет може да предлага много изпити, но всеки изпит се предлага точно от един факултет. За всеки кандидат-студент се пази и информация за оценка — дробно число от 2.0 до 6.0, което кандидат-студентът е получил за всеки изпит, на който се е явил (може да бъде `NULL` в случай, че кандидат-студентът не се е явил на изпит).

**Задание**:

а) Да се направи E/R модел на БД, която съхранява гореописаната информация. Начертайте 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 системата за кандидат-студентски изпити, описана в Задача 5, разглеждаме процес по добавяне на оценки от изпит в системата. Оценки могат да се добавят само в определен период след провеждането на изпита. След добавяне на оценка съответният кандидат-студент трябва да бъде известѐн. След потвърждаване на въвеждането на всички оценки съответният изпит се обявява за оценен и може да започне процес по класиране за специалностите, за които няма неоценени изпити.

Да се опише пълно потребителски случай за процеса по добавяне на оценка. Да се включат поне един алтернативен и един неуспешен сценарий. В описанието да се включат като минимум: актьори, които участват; предусловия; постусловия; взаимодействие с други потребителски случаи (ако е приложимо, но само с препратка). Посочете допълнителни нефункционални изисквания към този случай.

Задача 7. В контекста на системата, описана в Задачи 5 и 6, да се начертае UML диаграма на състоянията (State Machine Diagram), показваща състоянията на кандидат-студентски изпит. Да се изобразят поне 4 състояния и да се опишат условията за преходите между тях. За всеки преход, който е функционален, да се посочи името и да се опише кратко потребителски случай, чрез който се осъществява.

Добавете кратък текст, поясняващ предположенията, които сте направили за системата.

Задача 8. Пресметнете интеграла $\int_0^{\frac{\pi}{2}} \frac{dx}{2+\sin x}$.

**Чернова**