

СОФИЙСКИ УНИВЕРСИТЕТ
„СВ. КЛИМЕНТ ОХРИДСКИ“ФАКУЛТЕТ ПО МАТЕМАТИКА
И ИНФОРМАТИКА**ДЪРЖАВЕН ИЗПИТ**
ЗА ПОЛУЧАВАНЕ НА ОКС “БАКАЛАВЪР ПО ИНФОРМАТИКА”**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:

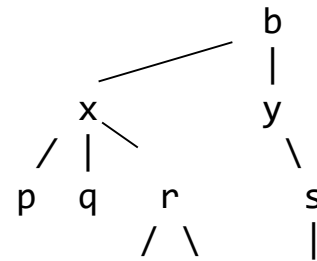
При подаден масив {13,14,7,2018,9,0}, след изпълнение на функцията **sortLex** масивът ще бъде подреден по следния начин: {0,13,14,2018,7,9}.

Задача 2. Задачата да се реши на един от езиците C/C++ или Java. В началото на решението си посочете кой език сте избрали.

Троично дърво от латински букви ще наричаме следната структура:

- Празното дърво е троично дърво от латински букви;
- Ако T_1 , T_2 и T_3 са троични дървета от латински букви, а x е латинска буква, то наредената четворка $\langle x, T_1, T_2, T_3 \rangle$ също е троично дърво от латински букви.

а) Да се напише функция **readLast**, която по дадено троично дърво от латински букви намира думата, която може да се прочете отляво надясно в последното му ниво. Ниво на троично дърво от латински букви наричаме списък от възли в дървото, които са равноотдалечени от корена.



Пример: за дървото вдясно функцията **readLast** трябва да връща думата “cat”.

Представянето на дървото е по Ваш избор. Опишете избраното от Вас представяне.

б) Да се напише функция **serialize**, която по дадени: (1) троично дърво от латински букви и (2) низ, описващ път до файл, записва дървото в текстов файл. Текстовото представяне на троично дърво да бъде следното:

- Текстовото представяне на празното дърво е звездичка (“*”);
- Текстовото представяне на троичното дърво $\langle x, T_1, T_2, T_3 \rangle$ е “(x T1 T2 T3)”, където T_1 , T_2 и T_3 са текстовите представяния съответно на T_1 , T_2 и T_3 .

Пример: Текстовото представяне на дървото горе е:

(b (x (p * * *) (q * * *) (r (c * * *) * (a * * *))) (y * * (s * (t * * * *) *) *)

За реализацията на функциите **readLast** и **serialize** е позволено използването на стандартните за съответния език библиотечни функции.

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 **f1** и непразен списък от числа **x1**. 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 **f1**, които имат неподвижна точка сред числата в **x1**, функцията **sumMinFix** да намира сумата на най-малките им такива неподвижни точки. Ако никоя функция от **f1** няма неподвижна точка сред числата в **x1**, 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 R⁵RS за *Scheme* и *Prelude* за *Haskell*.

Scheme

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

(define (sumMinFix f1 x1)
  (_____
    (_____
      (lambda (f)
        (apply _____
          (addDefault _____
            (_____
              (lambda (x) _____ ) x1)))) f1))))
```

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

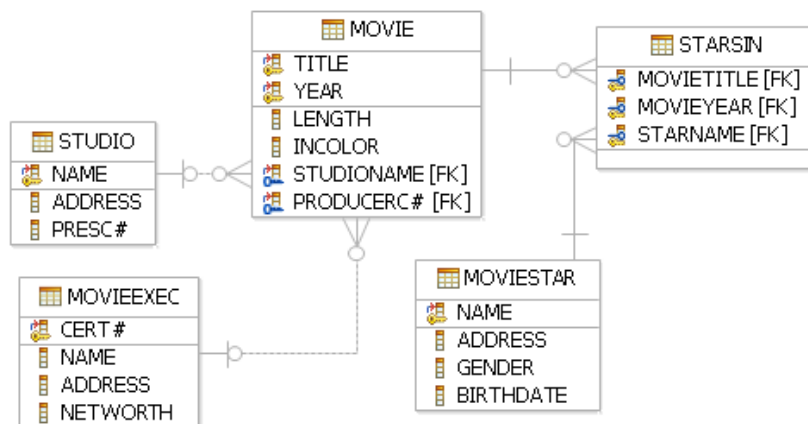
sumMinFix f1 x1 =
  _____
  (_____
    (\f -> _____
      (addDefault _____
        [ _____ | x <- x1, _____ ])) f1)
```

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. Файлът **f1** се намира в текущата директория и има следното съдържание:

1 3 5 2 4 6

Текстов файл с име **procA** съдържа зададената по-долу последователност от команди на bash за Linux. Напишете вдясно какво ще бъде изведено на стандартния изход и какво ще е съдържанието на файла **f2** след стартиране на **procA** с команден ред

bash procA ab cd ef gh

ако на стандартния вход бъде подадена последователността от символи **8 3**

```
var=3
for i
do for j in `cat f1`
do if test $j -lt $#
then echo $i $j
var=`expr $var \* $j`
echo $var $i > f2
else var=`expr $var + $j`
echo $var $j >> f2
fi
done
break
done
read s1 s2
until grep $s1 f2
do set $s2 $var $#
wc -c f2
shift
echo $1 $2
exit
cat f2
done
set `head -21 f2`
echo $*
shift
echo $2
```

Задача 6. 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 индексирание, 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 7. Разглеждаме обикновени графи. *Хроматично число* на граф $G = (V, E)$ е минималният брой цветове, с които може да се оцветят върховете на графа по такъв начин, че за всяко ребро (u, v) , краищата му u и v са в различни цветове. Хроматичното число на G се бележи с $\chi(G)$.

Докажете, че за всеки граф G е вярно, че $\chi(G) \leq \frac{1}{2} + \sqrt{2m + \frac{1}{4}}$, където m е броят на ребрата на G .

Упътване: Имайте предвид, че върховете от всеки цвят са антиклика (с други думи, независимо множество). Иначе казано, оцветяването на върховете е разбиване на множеството от върховете на минимален брой антиклики. Докажете, че за всеки две от тези антиклики, между поне един връх от едната и поне един връх от другата трябва да има ребро. Какво следва от това?

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

Чернова