СОФИЙСКИ УНИВЕРСИТЕТ
“СВ. КЛИМЕНТ ОХРИДСКИ”


ФАКУЛТЕТ ПО МАТЕМАТИКА
И ИНФОРМАТИКА

# ДЪРЖАВЕН ИЗПИТ

## ЗА ПОЛУЧАВАНЕ НА ОКС “БАКАЛАВЪР ПО КОМПЮТЪРНИ НАУКИ”

## 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.** *Задачата да се реши на езика C/C++.*

Да се попълнят празните места в кода на функциите така, че те да отговарят на описанието си.

```
// A) find използва алгоритъма за двоично търсене (binary search), за да
// провери дали value се съдържа в масива arr, състоящ се от size елемента.
// Функцията връща true ако това е така и false в противен случай.

bool find(int value, int* arr, size_t size)
{
    if (size == 0) return ______________;

    size_t mid = size / 2;
    if (value == arr[mid]) return ______________;

    if (value < arr[mid])
        return find(______________, ____________________, ______________);
    else
        return find(______________, arr + ______________, ______________);
}

// Б) fold_left изпълнява ляво свиване (left fold) върху масива arr, съдържащ size елемента,
// прилагайки операцията op. Началната стойност е nil.
// Функцията връща стойността op(...op(op(nil, a[0]), a[1]), ..., a[size-1]).

template <typename ReturnType, typename InputType, typename OpType>
ReturnType fold_left(InputType* arr, size_t size, OpType op, ReturnType nil)
{
   ______________ result = ______________;

   for (size_t i = 0; i < ______________; ++i)
       result = op(______________, ______________);

   return result;
}

int op(char Digit, int Result)
{
   return (______________ * 10) + (______________ - '0');
}

// Преобразува символен низ от десетични цифри до величина от тип int
int str_to_int(const char * str)
{
   return (str == nullptr) ? 0 : fold_left(str, ____________________, op, ______________);
}
```

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

*Дърво с етикети* ще наричаме кореново дърво $T = (V, E, r)$ с множество от върхове $V$, множество от ребра $E$ и корен $r$, за което са дефинирани две допълнителни функции:

$$value : E \longrightarrow \{n \in \mathbb{N} \mid 0 \leq n < 2^{32}\} \quad \text{и} \quad label : E \longrightarrow \{a, b, \ldots, z\}.$$

*Клон* в $T$ ще наричаме път $\pi = (v_0, v_1, \ldots, v_n)$, за който $v_n$ е листо на $T$ и $v_i$ е родител на $v_{i+1}$ за всяко $i < n$.

За всеки клон $\pi = (v_0, v_1, \ldots, v_n)$ дефинираме:

$$val(\pi) = \sum_{i=0}^{n-1} value(\langle v_i, v_{i+1} \rangle) \quad \text{и} \quad word(\pi) = label(\langle v_0, v_1 \rangle) label(\langle v_1, v_2 \rangle) \ldots label(\langle v_{n-1}, v_n \rangle).$$

А) Да се избере, дефинира и опише подходящо представяне на дърво от описания вид.

Б) За така дефинираното представяне да се реализира функцията:

```
CommonBranches(<labeled tree> T, <vertex> u, <vertex> v, <integer> k),
```

която по дадено дърво с етикети $T = (V, E, r)$, два негови върха $u$ и $v$ и естествено число $k$ извежда на стандартния изход всички думи $w$, за които има клони $\pi_u$ и $\pi_v$ с начало $u$ и съответно $v$, за които $word(\pi_u) = word(\pi_v) = w$ и $val(\pi_u) + val(\pi_v) = k$.

***Забележки:***

- *Отделните думи да са разделени със символа за нов ред. Една дума може да се извежда повече от веднъж.*
- *Функционалности на структурата дърво с етикети, които нямат отношение към задачата и не се използват във функцията `CommonBranches`, няма да бъдат оценявани.*

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

"Етикет" наричаме наредена двойка от низове — име и стойност, а "анотирана данна" наричаме наредена двойка от данна и списък от етикети за нея. Разглеждаме база данни, представена като списък от анотирани низове. "Анотатор" наричаме функция, която приема данна (низ) и връща списък от етикети за нея. Да се попълнят по подходящ начин празните полета по-долу, така че при подаване на база данни `db` и списък от анотатори `annotators`, функцията `annotate` да връща актуализирана база данни, в която за всяка данна в `db` са добавени етикетите, върнати за нея от анотаторите в `annotators`. Ако даден етикет вече съществува за дадена данна, той да не се добавя повторно. Да се реализира помощната функция `addIfNew`, така че да добавя елемента в `x` в началото на списъка `l` само ако `x` не се среща вече в `l`.

***Упътване:*** *могат да се използват наготово функциите `append`, `apply`, `concat`, `concatMap`, `elem`, `filter`, `foldr`, `map`, `member`, `sum` и стандартните функции в R5RS за Scheme и в Prelude за Haskell.*

## Scheme

```
(define (addIfNew x l) ____________________________________________)
(define (annotate db annotators)
  (____________
    (lambda (item-labels-pair)
      (let ((item (car item-labels-pair)) (labels (cdr item-labels-pair)))
        (cons item (____________ addIfNew labels
                     (_____________________________
                       (lambda (annotator) _____________________________) annotators))))) db))
```

***Пример:***

```
(define db (list (cons "scheme" (list (cons "typing" "dynamic") (cons "evaluation" "strict")))
                 (cons "haskell" (list (cons "typing" "static"))) (cons "c++" (list))))
(define (evaluation lang)
  (case lang (("scheme")  (list (cons "evaluation" "strict") (cons "macros" "true")))
             (("haskell") (list (cons "evaluation" "lazy"))) (("c++") (evaluation "scheme"))))
(define (purity lang) (if (eqv? lang "haskell") (list (cons "pure" "true")) (list)))
(annotate db (list evaluation purity)) ⟶
  (("scheme" ("macros" . "true") ("typing" . "dynamic") ("evaluation" . "strict"))
   ("haskell" ("evaluation" . "lazy") ("pure" . "true") ("typing" . "static"))
   ("c++" ("evaluation" . "strict") ("macros" . "true")))
```

## Haskell

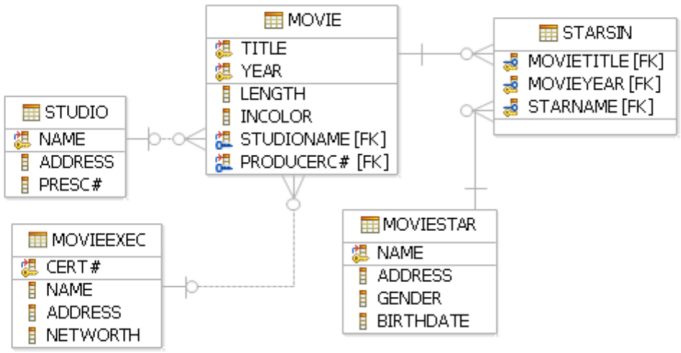
```
addIfNew x l = ____________________________________________
annotate db annotators =

  ____________
    (\(item, labels) ->
      (item, ____________ addIfNew labels
               (_____________________________
                 (\annotator -> _____________________________) annotators))) db
```

***Пример:***

```
db = [("scheme", [("typing", "dynamic"), ("evaluation", "strict")]),
      ("haskell", [("typing", "static")]), ("c++", [])]
evaluation "scheme"  = [("evaluation", "strict"), ("macros", "true")]
evaluation "haskell" = [("evaluation", "lazy")]
evaluation "c++"     = evaluation "scheme"
purity lang = if lang == "haskell" then [("pure", "true")] else []
annotate db [evaluation, purity] ⟶
  [ ("scheme",  [ ("macros", "true"), ("typing", "dynamic"), ("evaluation", "strict") ] ),
    ("haskell", [ ("evaluation", "lazy"), ("pure", "true"), ("typing", "static") ] ),
    ("c++", [ ("evaluation", "strict"), ("macros", "true") ] ) ]
```

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




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 `MovieExec` съдържа информация за продуцентите на филми.

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

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 черно-бял
- `studioname` — име на студио, външен ключ към `Studio.name`;
- `producerc#` — номер на сертификат на продуцента, външен ключ към `MovieExec.cert#`.

Таблицата `Studio` съдържа информация за филмови студиа:

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

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.

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

**Зад 1.** Да се огради буквата на заявката, която извежда за всеки продуцент името му и броя на филмите му по години. Продуценти, които нямат нито един филм, НЕ трябва да присъстват в резултатното множество.

А)
```
SELECT ME.NAME, M.YEAR, COUNT(*) AS CNT
FROM MOVIEEXEC ME LEFT JOIN MOVIE M
ON ME.CERT# = M.PRODUCERC#
WHERE M.TITLE IS NULL
GROUP BY ME.CERT#, ME.NAME, M.YEAR;
```

Б)
```
SELECT ME.NAME, M.YEAR, COUNT(*) AS CNT
FROM MOVIEEXEC ME
JOIN MOVIE M
ON ME.CERT# = M.PRODUCERC#
GROUP BY ME.CERT#, ME.NAME, M.YEAR;
```

В)
```
SELECT ME.NAME, M.YEAR, COUNT(*) AS CNT
FROM MOVIEEXEC ME, MOVIE M
GROUP BY ME.CERT#, ME.NAME, M.YEAR
WHERE ME.CERT# = M.PRODUCERC#;
```

Г)
```
SELECT ME.NAME, M.YEAR, COUNT(*) AS CNT
FROM MOVIEEXEC ME
JOIN MOVIE M
ON ME.CERT# = M.PRODUCERC#
ORDER BY ME.CERT#, ME.NAME, M.YEAR;
```

**Зад 2.** Да се напише заявка, която да изведе името на най-младата звезда (полът е без значение).

**Задача 5.** Числата на Фибоначи се дефинират чрез следното рекурентно уравнение:

$$F_n = \begin{cases} 0, & \text{ако } n = 0, \\ 1, & \text{ако } n = 1, \\ F_{n-1} + F_{n-2}, & \text{ако } n > 1. \end{cases}$$

Да се докаже, че

$$\forall n \geq 1 \,:\, F_{2n+1} = F_n^2 + F_{n+1}^2$$

***Упътване:*** *задачата може да се реши с комбинаторни разсъждения. $F_m$ за $m \geq 1$ е броят на различните начини да се изкачи стълба с $m-1$ стъпала, ако на всяка крачка качваме едно или две стъпала. Това може да се ползва наготово.*

*Примерно, стълба с 3 стъпала може да се изкачи по три начина, ако на всяка крачка качваме едно или две стъпала: или стъпало по стъпало, или първо две стъпала наведнъж и после едно стъпало, или първо едно стъпало и после две стъпала наведнъж. Броят на начините е 3, а 3 е точно $F_4$.*

*Виждаме, че $F_{2n+1}$ е броят на начините да се качи стълба с $2n$ стъпала, ако на всяка крачка качваме едно или две стъпала. Разбийте тези качвания по това, дали се стъпва върху $n$-тото стъпало, или не.*

**Задача 6.** Даден е граф $G(V, E)$ с $|V| = n$ върха. Някои от ребрата му са ориентирани, други не. Ориентираните ребра в G не образуват цикли.

Възможно ли е да се зададе ориентация на неориентираните ребра, без да се появят ориентирани цикли в $G$? Да се предложи бърз алгоритъм, който задава такава ориентация.

**Задача 7.** Да се пресметне интегралът $\int_0^2 ln(x^2 + 4)\, dx$.

## Чернова