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. *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 езика C++.*

А) Да се посочи какво ще се изведе при изпълнението на всеки от номерираните редове във функцията `main()` (възможно е да има ред, който не извежда нищо).

```
class Base {
public:
    Base()           { std::cout << "Base::Base\n"; }
    ~Base()          { std::cout << "Base::~Base\n"; }
    virtual void f() { std::cout << "Base::f\n"; }
};

class Derived : public Base {
public:
    Derived()        { std::cout << "Derived::Derived\n"; }
    ~Derived()       { std::cout << "Derived::~Derived\n"; }
    virtual void f() { std::cout << "Derived::f\n"; }
};

int main() {
1:      Base *p = new Derived();

2:      p->f();

3:      delete p;

4:      Derived d;

5:      Base b = d;

6:      b.f();

7:  } // излизане от main
```

Б) За дадения по-долу фрагмент, за всеки от редовете по-долу да се обясни какво прави той и какво ще се случи при неговото изпълнение. В случай, че изпълнението на някой от редовете ще предизвика грешка, да се обясни каква и защо.

```
Base *arr = new Base[3];

delete[] arr;

delete[] arr;
```

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

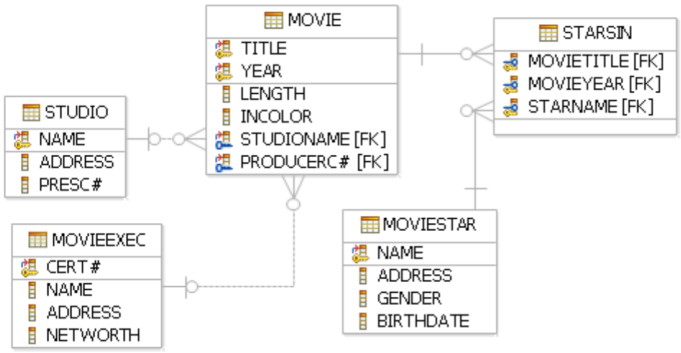
```
addIfNew x l = ____________________________________________
annotate db annotators =

  _____________
    (\(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. Информационна система съхранява информация за пациентите на стоматологична клиника. Съхранява се информация за пациента: номер на пациент — цяло число, уникално за всеки пациент на клиниката, и име на пациент — низ до 50 символа. Всеки пациент си записва час при стоматолог с определени оплаквания. Съхранява се информация за стоматолога: номер на стоматолог — цяло число уникално за всеки стоматолог в клиниката, и име на стоматолог — низ до 50 символа. В системата се пази информация и за процедурите, които стоматолог може да извършва на пациент. За всяка процедура се пази информация за: номер на процедура — цяло число, описание на процедурата – низ до 100 символа и цена на процедурата — дробно число. В базата от данни трябва да се пази и информация за часа и датата, когато пациентът се е записал за стоматолог, и с какви оплаквания се е записал — низ до 50 символа. В сила са следните ограничения:

- В една и съща дата и час един пациент не може да бъде записан за повече от един стоматолог.
- В една и съща дата и час един стоматолог не може да има повече от един пациент.
- В рамките на един записан час, стоматолог може да направи повече от една процедура на пациент (например: поставяне на упойка, поставяне на пломба и т.н.).

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. А) В контекста на информационната система за стоматологична клиника, представена в задача 5 (за E/R моделиране), разглеждаме процес по записване на час за преглед от пациент. Може да се запише час при определен специалист или да не се посочи такъв, ако няма предпочитания. Записаният час може да бъде в рамките на следващите 90 календарни дни, само в работно време за съответния специалист.

Да се опише в пълен формат потребителски случай "Записване на час за преглед". В описанието да се включат минимум:

- актьори, които участват;
- предусловия; следусловия (резултати);
- основен успешен сценарий;
- алтернативен сценарий;
- неуспешен сценарий.

При описанието да се посочат също взаимодействие с други потребителски случаи (ако е приложимо, но само с препратка) и какви проверки за коректност се извършват при обработката. Да се посочат допълнителни нефункционални изисквания към този потребителски случай. Ако са направени някакви допускания във връзка с представения сценарий, те да се опишат явно в решението.

Б) В контекста на информационната система за стоматологична клиника, представена в задача 5 (за E/R моделиране) и описанието на потребителския случай, изготвено в точка **А** (описание на потребителски случай в пълен формат), да се подготви диаграма на последователността (sequence diagram) в термините на UML стандарта. Да се опише значението на елементите от нотацията, които са използвани.

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

Чернова