

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 защити 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 защитават в края на работата.
- Времето за работа по изпита е 3 часа.

*Изпитната комисия ви пожелава успешна работа!*

**Задача 1.** Да се дефинира рекурсивна функция на езика C++

```
void print_backwards(const char* begin, const char* end).
```

Параметрите `begin` и `end` са указатели към елементи на буфер от символи, като  $end \geq begin$ . Символен низ с начало `begin` и край `end` наричаме последователността от символи, започваща със символа, намиращ се на адрес `begin` и завършваща със символа на адрес `end-1`, включително, дори и ако някой от тези символи е с код 0. Дума в такъв символен низ наричаме всяка непразна подпоследователност от стандартни символи (с код  $> 32$ ), ограничена от двете страни или от някой от краищата на низа, или от специален символ (с код  $\leq 32$ ).

Функцията `print_backwards` да извежда на стандартния изход всички думи в низа с начало `begin` и край `end` в ред, обратен на срещането им в низа, и разделени с точно един интервал.

Да се дефинира функция

```
void print_backwards(const char* string),
```

която прилага рекурсивната функция `print_backwards` върху символния низ `string`.

Да се демонстрира извикването на последната функция в кратка програма.

Пример: Извикването `print_backwards("I\tneed a break!");` извежда низа "break! a need I".

---

**Задача 2.** а) Нека  $a$  и  $d$  са стойности от произволен тип  $T$ , поддържащ операциите събиране и сравнение за равенство, а  $n$  е цяло положително число.

Да се дефинира шаблон на клас `Sequence`, описващ редицата  $(a, a + d, \dots, a + n * d)$  по зададени чрез конструктора на класа стойности на  $a$ ,  $d$  и  $n$ , където  $a + n * d$  е означено  $n$ -кратното събиране на  $d$  към  $a$ . Чрез обекти от клас `Sequence<T>` да може да се обхождат последователно елементите на дадена редица с помощта на следните методи:

1. Метод `current` за получаване на текущия елемент на редицата.
2. Метод `next` за преминаване към следващия елемент на редицата.
3. Булев метод `last` за проверка дали е достигнат последния елемент на редицата.

Обхождането на редицата започва с първия ѝ елемент. Всяко изпълнение на метода `next` води до преминаване към следващия елемент в редицата. По този начин, ако методът `next` е изпълнен точно  $i$  ( $0 \leq i \leq n$ ) пъти, чрез метода `current` ще бъде достъпен елементът  $a + i * d$ . Ако обхождането на редицата вече е достигнало последния ѝ елемент, методът `next` да няма ефект.

Пример: Следният програмен фрагмент

```
Sequence<int> s(0, 1, 10);
std::cout << s.current() << std::endl;
do {
    s.next();
    std::cout << s.current() << std::endl;
} while (!s.last());
```

ще изведе числата 0, 1, ..., 10.

б) Да се дефинира шаблон на клас `MappedSequence`, наследник на `Sequence`. Освен  $a$ ,  $d$  и  $n$ , конструкторът на клас `MappedSequence<T>` да получава и едноместна функция  $f : T \rightarrow T$ . Чрез този шаблон да може се обхождат последователно елементите на редицата  $(f(a), f(a + d), \dots, f(a + n * d))$  по описания в предишното подусловие начин.

Пример: Следният програмен фрагмент

```
int square (int x) { return x * x; }
int main ()
{
    MappedSequence<int> s(0, 1, 10, square);
    std::cout << s.current() << std::endl;
    do {
        s.next();
        std::cout << s.current() << std::endl;
    } while (!s.last());
    return 0;
}
```

ще изведе числата 0, 1, 4, ..., 100.

**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.

При планински преход GPS устройствата могат да записват “следа” на изминатия път във вид на поредица от поне две точки. Всяка точка се задава с наредена двойка от цели числа: изминато разстояние от началото на прехода и надморска височина. Точките в следата са подредени в строго нарастващ ред по изминатото разстояние. “Наклон” на участъка между две точки наричаме абсолютната стойност на отношението на разликата във височините им и дължината на изминатия път между тях.

Да се попълнят по подходящ начин празните полета в програмата по-долу, така че при извикване на функцията `easiestTrackUnder` с максимална дължина `maxLen` и списък от следи `tracks` да се връща следата, чиито максимален наклон е възможно най-малък от всички следи от `tracks`, които са по-къси от `maxLen`. Ако има повече от една следа, отговаряща на условието, е без значение коя от тях ще бъде върната като резултат. Ако няма нито една следа, отговаряща на условието, поведението на функцията е недефинирано (може да бъде произволно).

Да се реализират помощните функции `maxSlope`, която намира максималния наклон на участък в пътека и `argMin`, която намира елемента в непразния списък `l`, над който функцията `f` дава минимална стойност.

**Упътване:** могат да се използват наготово функциите `abs`, `apply`, `filter`, `foldr`, `foldr1`, `last`, `map`, `min`, `minimum`, `max`, `maximum`, `reverse`, `zip`, както и всички стандартни функции в R5RS за Scheme и в Prelude за Haskell.

### Scheme

```
(define (argMin f l) (_____ l))

(define (maxSlope track)
  (_____ (map (lambda (_____) _____)
                (reverse (cdr track)) (cdr (reverse track))))))

(define (easiestTrackUnder maxLen tracks)
  (argMin maxSlope (_____ (lambda (track) _____) tracks)))
```

#### Пример:

```
(define tracks '(((0 . 900) (100 . 910) (200 . 925) (300 . 905) (600 . 950))
                  ((0 . 1300) (100 . 1305) (500 . 1340) (800 . 1360) (1000 . 1320))
                  ((0 . 800) (200 . 830) (300 . 845) (600 . 880) (800 . 830))))

(maxSlope (car tracks)) → 0.2
(easiestTrackUnder 800 tracks) → ((0 . 900) (100 . 910) (200 . 925) (300 . 905) (600 . 950))
```

### Haskell

```
argMin f l = _____ l

maxSlope track = _____
                (map (\_____ -> _____)
                  (zip track (tail track)))

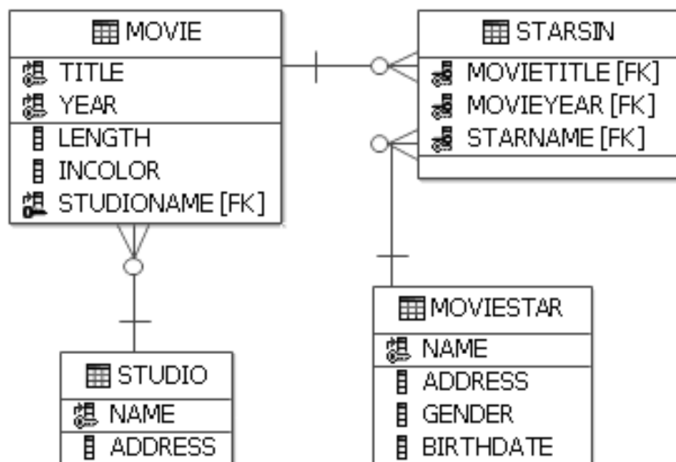
easiestTrackUnder maxLen tracks =
  argMin maxSlope (_____ (_____ tracks))
```

#### Пример:

```
tracks = [(0, 900), (100, 910), (200, 925), (300, 905), (600, 950)],
          [(0, 1300), (100, 1305), (500, 1340), (800, 1360), (1000, 1320)],
          [(0, 800), (200, 830), (300, 845), (600, 880), (800, 830)]

maxSlope (head tracks) → 0.2
easiestTrackUnder 800 tracks → [(0, 900), (100, 910), (200, 925), (300, 905), (600, 950)]
```

**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 студия:

- 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.

а) Да се напише заявка, която извежда имената и рождените дати на всички филмови звезди, чието име не съдържа "Jr." и са играли в поне един цветен филм. Първо да се изведат най-младите звезди, а звезди, родени на една и съща дата, да се изведат по азбучен ред.

б) Да се напише заявка, която извежда следната информация за всяка актриса, играла в най-много 6 филма:

- име;
- рождена година (напр. ако актрисата е родена на 1.1.1995 г., в колоната да пише 1995);
- брой различни студия, с които е работила.

Ако за дадена актриса няма информация в какви филми е играла, за нея също да се изведе ред с горната информация, като за брой студия се изведе 0.

**Задача 5.** Информационна система съхранява информация за агенция, която предлага услуги за ремонтни дейности. Съхранява се информация за клиентите на агенцията: телефонен номер на клиент – низ от 12 символа, уникален за всеки клиент, име на клиент – низ до 50 символа и e-mail – низ до 30 символа. Съхранява се информация и за майсторите, които извършват ремонтните услуги: телефонен номер на майстора – низ от 12 символа, уникален за всеки майстор, име на майстор – низ до 50 символа и e-mail – низ до 30 символа. Всеки майстор може да извършва една или повече услуги. За услугите се съхранява тип на услугата – низ до 40 символа, например: шпакловане, боядисване, редене на паркет и т.н. Всеки майстор изпълнява дадена услуга за определена сума за един час. Клиентът може да направи поръчка, която да включва една или повече услуги, като в системата се съхранява датата, когато е направена поръчката, статусът на поръчката и адресът, където трябва да бъде извършена услугата (услугите). Всяка поръчка еднозначно се определя от клиента и от датата на поръчката. По поръчката на клиента могат да работят един или повече майстори за изпълнението на една или повече услуги. Съхранява се информация и за датата, на която майсторът е работил по услуга от дадена клиентска поръчка.

**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 ремонтни дейности, представена в Задача 5 (за E/R моделиране), разглеждаме процес за подаване на поръчка на ремонтни дейности от клиент. Клиентът може да подаде поръчка чрез системата или по телефона. Ако е подадена по телефона, служител на фирмата подготвя поръчка в системата. Ако има предпочитания, клиентът може да посочи един или повече майстори за изпълнение на поръчката. Поръчката се приема за изпълнение, след като се заплати капаро в размер на минимум 10% от стойността на услугите, включени в нея. Клиентът може да се откаже от поръчката, без задължение за други плащания, ако нейното изпълнение не е започнало. Ако е започнало, следва да заплати за извършената работа. Поръчката е завършена, ако е одобрена от клиента и стойността на извършените услуги е заплатена в пълен размер.

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 опише в пълен формат потребителски случай за подаване на поръчка за ремонтни дейности. 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.
- б) Да се подготви диаграма на състоянията на обект **Поръчка** в термините на UML стандарта. Да се добави кратко текстово описание на диаграмата.

**Задача 7.** В урна има 5 бели, 7 зелени и 3 червени топки. На всеки опит вадим от урната едновременно две топки, записваме цвета им, след което връщаме топките обратно в урната. Дефинираме събитие

$$A = \{\text{Изтеглени са една бяла и една зелена топка}\}.$$

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

б) Нека  $X$  е броят на сбъдванията на събитието  $A$  при провеждане на 5 опита, да се пресметнат:  $P(X = 3)$ , математическото очакване  $EX$  и дисперсията  $DX$ .

в) Нека белите топки са 5, зелените 7, но броят на червените е  $Z$ . Каква трябва да бъде стойността на  $Z$ , така че средният брой на неуспешните опити до първото сбъждане на събитието  $A$  да бъде точно пет? Отговорът да се обоснове.



**Чернова**