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. 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` ≥ `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 ≤ 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. Цикличен едносвързан списък наричаме линеен едносвързан списък, в който указателят за следващ елемент на последния елемент сочи към първия елемент, вместо да е нулев. Да се реализира функция `void unite(Node* list)`, която получава като параметър указател към елемент на цикличен едносвързан списък от символни низове, описан от следната структура:

```
struct Node {
    std::string text;
    Node* next;
};
```

Функцията да обединява всички двойки последователни елементи на списъка, за които последният символ на единия елемент съвпада с първия символ на непосредствено следващия го елемент, в общ елемент, чийто низ е съставен от слепването на низовете на двата елемента, разделени с тире.

Да се демонстрира работата на функцията в кратка програма, която прочита низове от стандартния вход (по един на ред), добавя ги в съответния цикличен списък, подава го на функцията `unite` и извежда последователно елементите на променения списък на стандартния изход, започвайки от **лексикографски най-малкия** низ.

Пример:

Вход:

```
street
taxi
ink
dog
smile
eat
tall
pass
```

Изход:

```
dog
smile-eat-tall
pass-street-taxi-ink
```

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))
```

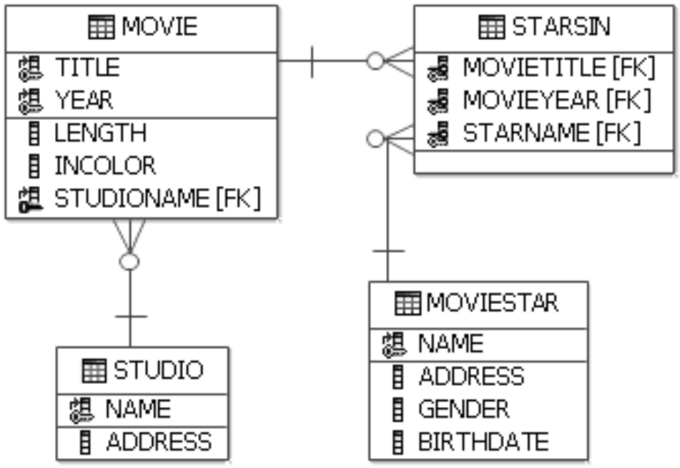
Haskell

```
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 студиа:

- `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 рожденните 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 студиа, 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 студиа се изведе 0.

Задача 5. Нека $G(V, E)$ е неориентиран свързан граф с поне два върха.
Да се докаже, че в G има върхове с еднаква степен.

Упътване: Ползвайте принципа на Дирихле.

Задача 6. Вярно ли е, че за всеки регулярен език $L \subseteq \{0,1\}^*$, езикът:

$$\{w^{|w|} \mid w \in L\}$$

е регулярен? Отговорът да се обоснове.

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: $\mathrm{P}(X = 3)$, математическото очакване $\mathrm{E}X$ и дисперсията $\mathrm{D}X$.

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 сбъдване на събитието A да бъде точно пет? Отговорът да се обоснове.

Чернова