

СОФИЙСКИ УНИВЕРСИТЕТ
“СВ. КЛИМЕНТ ОХРИДСКИ”



ФАКУЛТЕТ ПО МАТЕМАТИКА
И ИНФОРМАТИКА

ДЪРЖАВЕН ИЗПИТ

ЗА ПОЛУЧАВАНЕ НА ОКС “БАКАЛАВЪР ПО ИНФОРМАТИКА”

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. а) Имената на служители в дадена компания и на техните преки ръководители можем да представим с двумерен масив `const char* leaders[][2]`.

Например:

Служител	Ръководител
Иван Иванов	Мария Иванова
Мария Иванова	Иван Драганов
Иван Драганов	Стоян Петров

Казваме, че служителят A е ръководител на служителя B , ако A е пряк ръководител на B или е пряк ръководител на ръководител на B . Да се дефинира рекурсивна функция:

```
bool is_subordinate (const char* employee,  
                    const char* manager,  
                    const char* leaders[][2],  
                    size_t n),
```

която проверява дали служителят с име `employee` е подчинен на служителя с име `manager` в компанията, описана с масива `leaders` с `n` реда.

б) Да се дефинира функция

```
const char* the_big_boss(const char* leaders[][2], size_t n),
```

намираща името на служителя, който се намира най-високо в йерархията на компанията, описана с масива `leaders` с `n` реда.

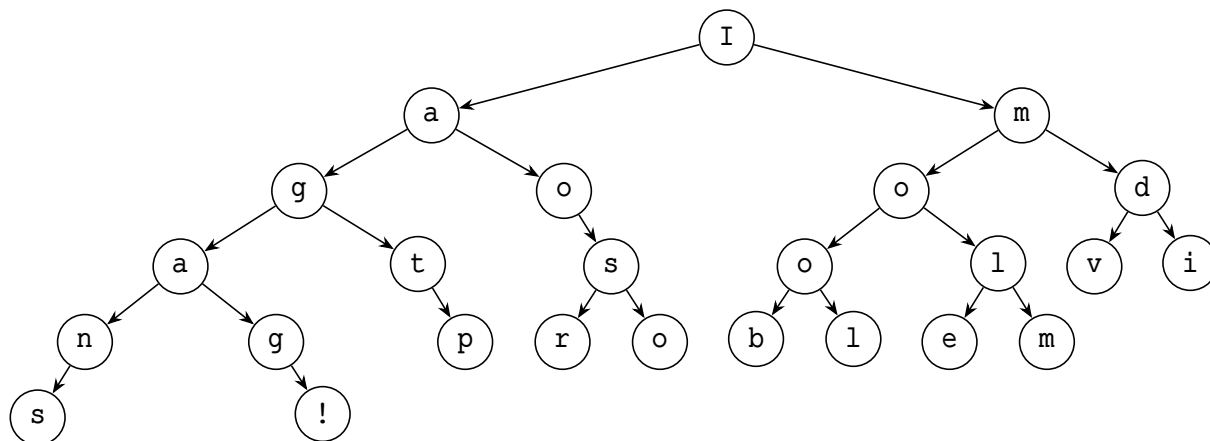
Приемаме, че йерархията от служителите, описана в `leaders` е коректна, т.е. всеки служител има по точно един пряк ръководител, с изключение на точно един служител, който няма пряк ръководител, и няма двама служители такива, че всеки от тях е ръководител на другия.

Да се демонстрира извикването на функциите с кратка програма.

Задача 2. Разглеждаме **двоично дърво** с данни от тип символ и елементи, описвани от следната структура:

```
struct Node {  
    char text;  
    Node *left, *right;  
};
```

Стойностите са малки и главни латински букви, цифри и препинателни знаци. Няма празни символи (интервал, нов ред, табулация) и символи от разширената ASCII таблица (с код по-голям от 127). Пример за такова дърво е:



Да се реализира функция `printText`, която получава като аргумент указател към корен на такова дърво и извежда на стандартния изход текста, записан във върховете, спазвайки следните правила:

- съдържанието на всяко ниво (елементи с еднаква дълбочина) се извежда на отделен ред;
- ако на дадено място в нивото липсва възел, да се изведе интервал;
- дървото се извежда до последния елемент от последното ниво.

За показаното като пример дърво трябва да се изведе следния текст:

```
|I|  
|am|  
|good|  
|at_solvi|  
|ng_p__roblem____|  
|s__!|
```

Забележка: за яснота на примера в началото и края на всеки ред е добавен символът `pipe` `|`, който `printText` не трябва да извежда.

Да се демонстрира работата на тази функция в кратка програма.

Забележка: позволено е използването на класовете контейнери от стандартната библиотека `STL`.

Задача 3. Задачата да се реши на един от езиците Scheme или Haskell.

Музикално произведение се описва с наредена тройка от име на автор, заглавие на произведението и дължина в секунди. Даден е непразен списък от музикални произведения `pl`, в който **те са подредени в нарастващ ред по дължина**. Препоръчваща функция наричаме функция, която приема наредена тройка, описваща текущо възпроизвеждащо се музикално произведение и връща заглавие на произведение, което се препоръчва да бъде следващото възпроизведено. Компанията *lify* е патентовала алгоритъм за препоръчваща функция, която за текущо възпроизвеждащо се произведение `p` с първи компонент (автор) `a` работи по следния начин за списък `pl`:

1. Препоръчва следващото по дължина произведение на `a` в `pl`, ако има такова.
2. В противен случай, препоръчва произведение с максимална дължина в `pl` из произведенията на всички автори, за които средната дължина на произведенията в `pl` е по-малка от средната дължина на произведенията на `a` в `pl`, ако има такива.
3. В противен случай, препоръчва следващото по дължина произведение в `pl` след `p`, а ако такова няма, препоръчва отново `p`.

Да се попълнят празните полета по-долу така, че функцията `recommender pl` да връща препоръчваща функция, реализираща алгоритъма на *lify*, описан по-горе, върху списък `pl`. Функцията `avgDuration` да връща средната дължина на произведенията на `author` в `pl`. Списъкът `option1` да съдържа произведенията на `a` с по-голяма дължина от `p`. Списъкът `option2` да съдържа произведенията на всички автори със средна дължина на произведенията по-малка от средната дължина на произведенията на `a`.

Упътване: могат да се използват наготово функциите `append`, `apply`, `filter`, `foldr`, `fromIntegral`, `length`, `list`, `last`, `map`, `reverse`, `sum`, както и всички функции в *R5RS* за Scheme и в *Prelude* за Haskell.

Scheme

```
(define (recommender pl)
  _____
  (define (avgDuration artist) _____)
  (define option1 _____)
  (define option2 _____)
  (if (not (null? option1)) _____ option1
      (if (not (null? option2)) _____ option2
          _____)))
```

Пример:

```
(define rf (recommender '(("Mozart" "The Marriage of Figaro Overture" 270) ("Gershwin"
  "Summertime" 300) ("Queen" "Bohemian Rhapsody" 355) ("Gershwin" "Rhapsody in Blue" 1100)))
(rf '("Mozart" "The Marriage of Figaro Overture" 270)) → "Summertime"
(rf '("Gershwin" "Summertime" 300)) → "Rhapsody in Blue"
(rf '("Gershwin" "Rhapsody in Blue" 1100)) → "Bohemian Rhapsody"
```

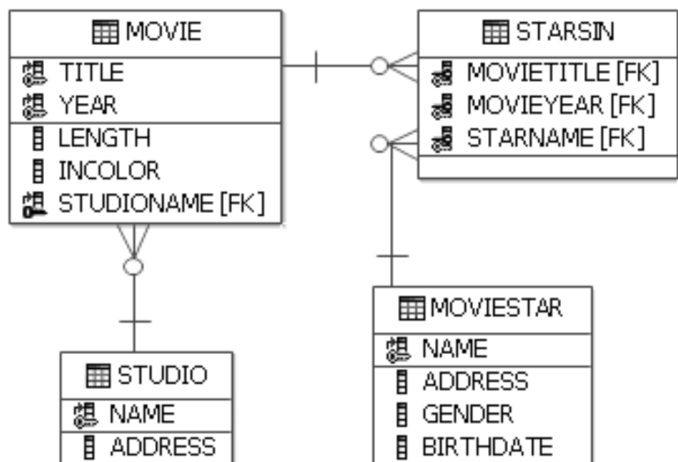
Haskell

```
recommender pl = _____
  let avgDuration artist = _____
      option1 = _____
      option2 = _____
  in if not (null option1) then (_____ option1 )
     else if not (null option2) then (_____ option2 )
     else _____
```

Пример:

```
rf = recommender [("Mozart","The Marriage of Figaro Overture",270), ("Gershwin","Summertime",
  300), ("Queen","Bohemian Rhapsody",355), ("Gershwin","Rhapsody in Blue",1100)]
rf ("Mozart","The Marriage of Figaro Overture",270) → "Summertime"
rf ("Gershwin", "Summertime", 300) → "Rhapsody in Blue"
rf ("Gershwin", "Rhapsody in Blue", 1100) → "Bohemian Rhapsody"
```

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.

Зад 1. Да се напише заявка, която извежда имената и адресите на всички студия, които имат поне един цветен и поне един черно-бял филм. Резултатът да се сортира възходящо по адрес.

Зад 2. Да се напише заявка, която за всяко студио с най-много три филма извежда:

- името му;
- адреса;
- средната дължина на филмите на това студио.

Студия без филми също да се изведат (за средна дължина да се извежда null или 0).

Задача 5. $G(V, E)$ е неориентиран граф. В него има цикъл със 7 върха.
Двуделен ли е графът G ? Дайте обоснован отговор.

Упътване: $G(V, E)$ е двуделен, когато множеството от върхове може да се разбие на две непразни, непресичащи се множества V_1 и V_2 , такива че всяко ребро свързва връх от V_1 с връх от V_2 .

Задача 6. За всеки два езика $L, M \subseteq \{0, 1\}^*$ означаваме

$$L^M = \{w \in L^{|u|} \mid u \in M\}.$$

Вярно ли е, че винаги, когато L и M са регулярни, то и L^M е регулярен език? Отговорът да се обоснове.

Задача 7. На всеки опит хвърляме три пъти последователно зар. Дефинираме събитие

$A = \{\text{точките при някое хвърляне са равни на сумата от точките на другите две хвърляния}\}.$

- а) Да се определи вероятността на A при извършване на един опит.
- б) Извършваме опити докато събитието A се изпълни. Нека X е броят на хвърлянията на зар, които сме направили при това. Да се намери математическото очакване EX и дисперсията DX .
- в) Колко опита трябва да бъдат направени, така че да е по-вероятно събитието A да се сбъдне поне веднъж, отколкото да не се сбъдне нито веднъж?

Чернова