СОФИЙСКИ УНИВЕРСИТЕТ
„СВ. КЛИМЕНТ ОХРИДСКИ“

ФАКУЛТЕТ ПО МАТЕМАТИКА
И ИНФОРМАТИКА

ДЪРЖАВЕН ИЗПИТ
ЗА ПОЛУЧАВАНЕ НА ОКС “БАКАЛАВЪР ПО КОМПЮТЪРНИ НАУКИ”

ЧАСТ I (ПРАКТИЧЕСКИ ЗАДАЧИ)
09.07.2019 г.

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++.*

Даден е двумерен масив с размер 6 на 6 от символи — малки и главни латински букви и цифри. Две клетки в него ще наричаме "съседни", ако имат обща стена (т.е. всяка клетка е съседна с най-много четири други, намиращи се под, над, вляво и вдясно от нея). Път с дължина N ще наричаме редица $a_0, a_1, \ldots a_{N-1}$ от клетки, за която:

1. `за всяко 0 ≤ i < N − 1 е изпълнено, че` a_i и a_{i+1} са съседни;
2. никоя от клетките не се среща повече от веднъж (т.е. няма цикли).

Да се попълнят празните места в кода на дадените по-долу функция `contains` и помощната ѝ функция `walk`. Функцията `contains` получава два аргумента – масив `arr` от дадения тип `char[6][6]` и символен низ `str`. Тя трябва да връща истина тогава и само тогава, когато в `arr` съществува път, чиито клетки образуват точно съдържанието на низа `str` (вижте примера по-долу). За определеност считаме, че функцията трябва да връща истина за празния низ.

Пример: За дадения по-долу двумерен масив `contains` трябва да върне истина, ако ѝ бъдат подадени низовете `"abcdefgh"`, `"A123B123C"` или `""`. За улеснение, за да може да ги видите по-лесно, те са маркирани в сиво.

y	u	f	a	b	c
G	o	p	g	B	1
c	b	a	h	3	2
d	k	j	i	2	3
e	f	Q	N	1	C
h	g	h	M	A	r

Кодът на двете функции е даден на следващия лист:

```
bool contains(char arr[6][6], const char* str)
{
```

```
    for (int row = 0; row < _____; _____________)
        for (int col = 0; col < _____; _____________)
            if (walk(arr, row, col, str))
                return _________;
    return _________;
}

bool walk(char arr[6][6], int row, int col, const char* str)
{
    if (*str == '\0')
        return ____________;

    if (row < 0 || col < 0 || row >= 6 || col >= 6)
        return ____________;

    if (arr[row][col] != *str)
        return ____________;

    arr[row][col] *= -1;

    bool result =
        walk(arr, row + ___, col,       str + 1) ||
        walk(arr, _________, _________, str + 1) ||
        walk(arr, _________, _________, str + 1) ||
        walk(arr, _________, _________, str + 1);

    arr[row][col] _________________;

    return result;
}
```

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

Разглеждаме кореново дърво, във възлите на което има записани двойки от символ (`char`) и цяло число (`int`). Всеки възел на дървото може да има произволен, краен брой наследници. За удобство разглеждаме функциите sym и val, дефинирани над множеството от възлите на дървото, така че за всеки възел v на дървото, в който е записана двойката $< a, b >$, $sym(v) = a$ и $val(v) = b$.

Клон в дървото `T` ще наричаме всеки път $\pi = (v_0, v_1, \ldots, v_n)$, за който $v_0, \ldots, v_n$ са върхове на `T`, v_n е листо на `T`, а v_i е родител на v_{i+1} за всяко $i < n$.
За всеки клон $\pi = (v_0, v_1, \ldots, v_n)$ на дървото `T` дефинираме съответни "дума" и "стойност" по следния начин:

$word(\pi) = sym(v_0)sym(v_1)\ldots sym(v_n)$,

$$value(\pi) = \sum_{i=0}^{n} val(v_i),$$

т.е. $word(\pi)$ е думата, която се получава от последователното прочитане на символите, записани във възлите на пътя, а $value(\pi)$ е сумата на числата, записани в тях.

а) Да се избере, дефинира и опише подходящо представяне на дърво от описания тип.
б) За така дефинираното представяне да се реализира функцията:

```
int sumVal ([подходящ тип] T, [подходящ тип] u, [подходящ тип] v),
```

която по дадено дърво `T` и два негови върха u и v намира и връща сумата от всички стойности $value(\pi_u) + value(\pi_v)$, за които π_u и π_v са клони с начала u и v (съответно) със свойството $word(\pi_u) = word(\pi_v)$. Ако такива клони няма, сумата се счита за 0.

Забележки:

1. *В зависимост от избраното представяне, параметърът `T` може да бъде пропуснат.*
2. *Не е нужно дефиницията на представянето на дървото да бъде пълна, нужно е само да е достатъчна за реализацията на функцията* ***`sumVal`****.*
3. *Позволено е използването на функциите и класовете от стандартната библиотека на избрания от Вас език.*

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

Големият онлайн магазин *Siberia* търси начин да увеличи продажбите като препоръчва на клиентите си подходящи продукти. За целта изследователският екип на *Siberia* експериментира с различни реализации на функция `bestFit`, която приема като параметър код на продукт `a` и връща код на друг продукт `b`, който клиентите на магазина най-вероятно биха си купили заедно с `a`. Задачата пред разработчиците на *Siberia* е да реализират функция `recommended`, която получава като параметри потребителска кошница `basket` (списък от целочислени кодове на продукти), функция `bestFit` и списък от продуктите на магазина `products` (списък от наредени двойки от уникален код на продукт и цена — неотрицателно число).

Да се попълнят по подходящ начин празните полета по-долу така, че функцията `recommended` да връща списък от кодовете на всички възможни препоръчани продукти. Допуска се в резултата някои кодове да се срещат повече от веднъж. Препоръчан продукт е такъв, който:

- все още не е в `basket`, но се получава като резултат от прилагането на функцията `bestFit` над някой от продуктите, които вече са в `basket`;
- има цена, която не надвишава общата цена на потребителската кошница, дефинирана като сумата от цените на продуктите в `basket`.

Помощните дефиниции `findPrice` и `basketCost` намират съответно цената на даден продукт `product` в списъка `products` и цената на потребителската кошница. Да се приеме, че `basket` съдържа само кодове на продукти в `products` и `bestFit` също връща само такива кодове.

Упътване: могат да се използват наготово функциите `apply`, `assoc`, `elem`, `filter`, `foldr`, `lookup`, `map`, `member`, `sum` и стандартните функции в R^{5}RS за Scheme и в Prelude за Haskell.

Scheme

```
(define (recommended basket bestFit products)
  (define (findPrice product)
    ____________________________________________________ products ____
  (define basketCost
    ____________________________________________________ basket ____

  (____________________
    (lambda (product)
      ____________________________________________________________)
        (____________________________________________ basket)))
```

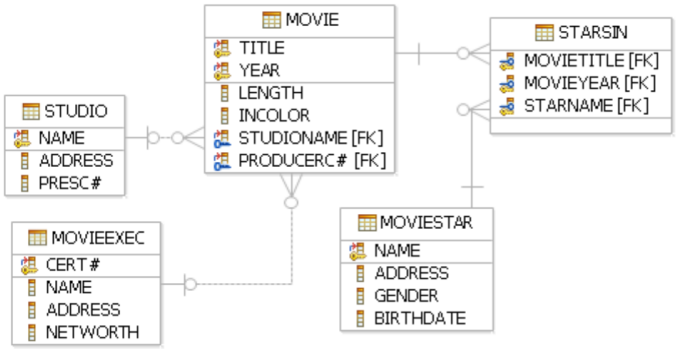
Haskell

```
recommended basket bestFit products =

  ____________________
  (\product ->
     ____________________________________________________________)
  (____________________________________________________ basket)

  where findPrice product =
          ____________________________________ products ____________________
        basketCost =
          ____________________________________________ basket ________
```

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 **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`;
- `producerc#` – номер на сертификат на продуцента, външен ключ към `MovieExec.cert#`.

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 **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 **Studio** съдържа информация за филмови студиа:

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

Таблицата **MovieExec** съдържа информация за продуцентите на филми.

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

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

а) Да се огради буквата на заявката, която извежда име на студио и броя на филмите му, за тези студия с по-малко от два филма. Студиата, които нямат нито един филм, **НЕ** трябва да присъстват в резултата.

A)
```
SELECT S.NAME, COUNT(M.TITLE) as CNT
FROM STUDIO S JOIN MOVIE M
ON S.NAME = M.STUDIONAME
GROUP BY S.NAME
HAVING CNT < 2;
```

B)
```
SELECT S.NAME, COUNT(M.TITLE) as CNT
FROM STUDIO S LEFT JOIN MOVIE M
ON S.NAME = M.STUDIONAME
WHERE M.TITLE IS NULL
GROUP BY S.NAME
HAVING COUNT(M.TITLE) < 2;
```

C)
```
SELECT S.NAME, COUNT(M.TITLE) as CNT
FROM STUDIO S JOIN MOVIE M
ON S.NAME = M.STUDIONAME
GROUP BY S.NAME
HAVING COUNT(M.TITLE) < 2;
```

D)
```
SELECT S.NAME, COUNT(M.TITLE) as CNT
FROM STUDIO S JOIN MOVIE M
ON S.NAME = M.STUDIONAME
WHERE COUNT(M.TITLE) < 2
GROUP BY S.NAME;
```

б) Да се напише заявка, която да изведе имената на всички продуценти с минимален нетен актив.

Задача 5. Текстовите файлове **x1** и **f1** се намират в текущата директория и имат следното съдържание:

x1

```
5 line five
.ne eight
: text
3456789
: check trial
.ne nine 10 line ten
lef
```

В текущата директория се намира също и празният файл **f4**. Текстов файл с име **comproc** съдържа зададената по-долу последователност от команди на **bash** за **Linux**. Да се напише какво ще бъде изведено на стандартния изход след еднократно стартиране на **comproc** със следния команден ред

```
bash comproc ab cd ef gh
```

и при подаване на последователността от символи **3 1** на стандартния вход.

```
grep `head  -1 f1` `tail -1 f1` | wc -l > f2
a=`cat f2`
echo $a $3
set  9  7  5  3
shift 2
for j in 1  2  3  4  5
do for i
   do if test $a -lt $i
      then cat f1 f2 > f3
           wc -l f3
           echo $i  $j  $a >> f4
      else tee f2 f3 < f1
           wc -w f2
           echo $i  $j  $a >> f4
      fi
   done
   echo $# >> f4
   break
done
read key1 key2
while cat f4 | grep $key2
do  sort f4
    a=`wc -c < f4`
    echo -n "Character count: $a"
    exit
done
grep $key1 f4
b=`wc -l < f4`
echo -n "Lines count: $b"
```

Задача 6. Нека Σ и Ω са две непразни и непресичащи се азбуки. За дума $w \in (\Sigma \cup \Omega)^*$ с $w_\Sigma \in \Sigma^*$ означаваме редицата от букви от Σ в реда, в който се срещат в w. Думата $w_\Omega \in \Omega^*$ се дефинира аналогично.

За езици $L_1 \subseteq \Sigma^*$и $L_2 \subseteq \Omega^*$с $L_1 \otimes L_2$означаваме езика:

$$L_1 \otimes L_2 = \{w \in (\Sigma \cup \Omega)^* \mid w_\Sigma \in L_1, w_\Omega \in L_2 \text{ и } |w_\Sigma| = |w_\Omega| \}$$

Винаги ли е вярно, че:

1. Ако L_1 е краен, то $L_1 \otimes L_2$ е регулярен?
2. Ако L_1 и L_2 са регулярни, то езикът $L_1 \otimes L_2$ е регулярен?

Отговорите да се обосноват. Отговор, който не е обоснован, се оценява с 0 точки.

Задача 7. Да се пресметне интегралът:

$$\int_1^8 \frac{dx}{\sqrt[3]{x}+x}$$

Чернова