СОФИЙСКИ УНИВЕРСИТЕТ “СВ. КЛИМЕНТ ОХРИДСКИ” | **ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА**

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

ЧАСТ I (ПРАКТИЧЕСКИ ЗАДАЧИ)
08.09.2010 г.

Време за работа – 3 часа

Драги абсолвенти,

Попълнете факултетния си номер на всички страници!

Решението на всяка от задачите се разполага само в мястото от края на условието на тази задача до началото на условието на следващата задача. Могат да се използват и двете стани на листата.

Изпитната комисия ви пожелава успешна работа.

Задача 1. (12 точки) Да се провери кои от следващите езици над азбуката $X = \{0, 1\}$ са едни и същи и кои са различни:

1. L_1 се представя чрез регулярния израз $(0 + 1)^* 0 (0 + 1)^* 0 (0 + 1)^*$
2. L_2 се разпознава от крайния детерминиран автомат $A = <\{q_0, q_1, q_2, q_3\},\{0,1\}, q_0, \delta, \{q_2, q_3\}>$ с функция на преходите δ, представена чрез таблицата долу вляво:

q	0	1
q_0	q_1	q_0
q_1	q_2	q_0
q_2	—	q_3
q_3	q_2	q_3

q	0	1
q_0	$\{q_1,q_2\}$	$\{q_0\}$
q_1	$\varnothing$	$\{q_0\}$
Q_2	$\{q_3\}$	$\varnothing$
Q_3	$\varnothing$	$\{q_3,q_4\}$
Q_4	$\{q_3\}$	$\varnothing$

3. L_3 се разпознава от крайния недетерминиран автомат $B = <\{q_0, q_1, q_2, q_3, q_4\},\{0,1\},q_0,\delta,\{q_3\}>$ с функция на преходите δ, представена чрез таблицата горе вдясно.
4. L_4 се разпознава от крайния детерминиран автомат $C = <\{q_0, q_1, q_2, q_3\}, \{0,1\}, q_0, \delta, \{q_3\}>$ с функция на преходите δ, представена чрез таблицата:

q	0	1
q_0	q_1	q_3
q_1	q_0	q_2
q_2	q_3	q_1
q_3	q_2	q_0

За да покажете, че два езика са различни посочете дума, която е от единия език, но не е от другия, а за да покажете, че два езика съвпадат сравнете крайните детерминирани автомати, които ги разпознават.

Задача 2. (13 точки) В текущия каталог се намира текстов файл `fileB.txt` със следното съдържание

```
12345$$6789
$$abcdefg
```

Да се напише вдясно на програмния код какво ще бъде изведено на стандартния изход (терминала) като резултат от изпълнението на файла, получен при успешна компилация на зададения по-долу програмен код на езика C, в който са използвани системни примитиви на ОС UNIX и LINUX:

```
#include <stdio.h>
#include <fcntl.h>
main()
{
   int fdr, fdw, n_byt, i = 0 , status;
   char buff [40], c ;

   if (fork( ))
   {
      Wait (&status);
      If (open("file_new", 0) != -1)
      execlp("grep", "grep", "$", "file_new",0);
   }
   else
   {
      if ((fdr = open("fileB.txt",0))== -1)
      { printf("\n Cannot open \n"); exit(1); }
      If (( fdw = creat ("file_new",0666))== -1)
      { printf ("\n Cannot creat \n"); exit (1); }
      n_byt = read (fdi,buff,40);
      c = buff[i++];
      if (c <= '1' || c >= '9')
      {
         while ( buff [i ++] != '\n' && i < n_byt)
            write( fdw, "$", 1 );
         write(fdw,"\n",1);
         write(1,"\n",1);
      }
      else { write(1,buff,n_byt );  write ( 1, "\n", 1 ); }
      write (fdw, "$", 1);
      close (fdr); close (fdw);
   }
 }
```

Задача 3. (12 точки) Дума в текст е всяка редица от малки и големи латински букви, цифри или символа '_'. Всички останали символи се считат за препинателни знаци. Да се напише функция, която преобразува текст по следния начин:

- редицата от думи в текста се обръща в обратен ред като първата дума става последна, втората дума става предпоследна и т.н., последната дума става първа.
- редицата от препинателни знаци в текста остава същата. Това означава, че ако има препинателен знак след първата дума на входния текст, то след преобразуването този препинателен знак е след последната дума от входния текст, която дума е първа в изходния текст.
- думите запазват реда на буквите си.
- препинателните знаци, ако са повече от един между две думи, също запазват реда си.

Примери.

Вход: `текст 1.` Изход: `1 текст.`	Входният текст съдържа две думи: "`текст`" и "`1`", а препинателните знаци са `' '` и `'.'`. В изхода думите сменят реда си, а препинателните знаци остават на местата си.
Вход: `текст !` Изход: `текст !`	Текстът "`текст !`" не се променя тъй като се състои само от една дума.
Вход: `1;;2;3;4;;5.` Изход: `5;;4;3;2;;1.`	Входният текст съдържа пет думи: "`1`", "`2`", "`3`", "`4`" и "`5`" и препинателните знаци: `';'` и `'.'`.

Задача 4. (12 точки) Да се дефинира клас `BinTree`, който представя двоично дърво с наредба на синовете и ориентация на ребрата от родител към дете, надписано със символи по върховете. **Ориентираният път** между два върха на дървото се представя с низа от надписите на последователните върхове на пътя.

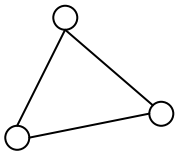
а) (2 т.) Да се дефинират член-данни и помощни структури за класа `BinTree`, които са необходими за избраното представяне.

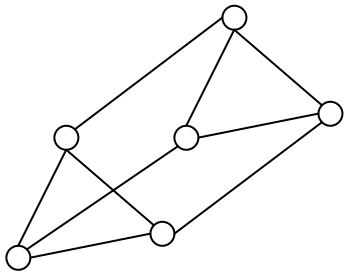
б) (3 т.) Да се реализира член-функция `isSymmetric()`, която проверява дали двоичното дърво е симетрично относно наредбата на синовете (ляво-дясно), т.е. дали съвпада с огледалния си образ.

в) (3 т.) Да се реализира член-функция `appendTree(t)` която вмъква дадено двоично дърво `t` на мястото на всички листа на дървото, чиито надписи съвпадат с надписа в корена на `t`.

г) (4 т.) Да се реализира член-функция `contains(s)`, която проверява дали даден низ `s` представя път в дървото.

Задача 5. (10 точки) Дадена е редицата от графи G_0, G_1, G_2, …, G_n, … Графът G_0 е пълният граф с три върха. Графът G_n, $n > 0$, е съставен от два подграфа G_{n-1}, като всеки връх на единия подграф е свързан с ребро с връх на другия подграф, а различните върхове на единия подграф са свързани с различни върхове на другия (на фигурата са показани графите и $G0$ и $G1$). Два алгоритъма са реализирани върху граф G_n от редицата, като единият е със сложност пропорционална на броя на върховете V_n, а другия на броя на ребрата E_n. Намерете асимптотична оценка $\Theta()$ на сложността на тези два алгоритъма. **Заб. Отговори, посочени без да са налице необходимите пресмятания няма да получат точки.**

G_0 

G_1 

Задача 6. (12 точки) Да се дефинира на езика Scheme функция `(generate-bin n)`, която генерира поток от всички естествени числа в интервала $[n, +\infty)$, представени чрез своя двоичен запис. Двоичният запис за едно число се представя като списък от нули и единици. Например:

- 1 → (1)
- 2 → (1 0)
- 10 → (1 0 1 0)

Примерни изпълнения:

(`generate-bin` 0) → Потокът е **`(0) (1) (1 0) (1 1) (1 0 0) (1 0 1) …`**

(`generate-bin` 10) → Потокът е **`(1 0 1 0) (1 0 1 1) (1 1 0 0) ( 1 1 0 1) …`**

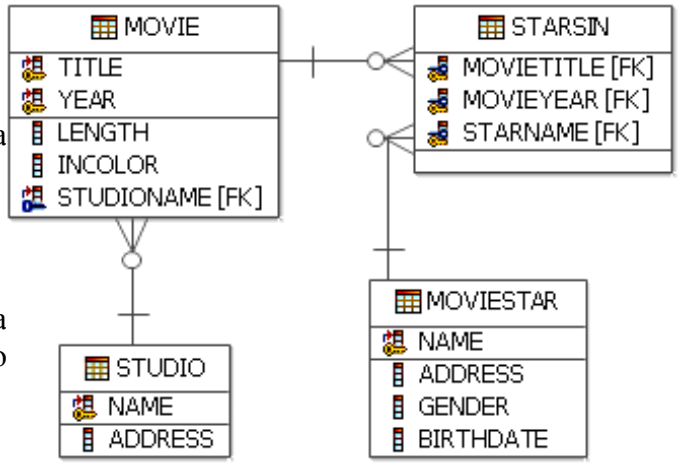
Задача 7. (13 точки) Нека $A = (a_{i,j})$ е матрица с m реда и n стълба. Да наречем **представяне** на A списъка $L_A = [\ [a_{11}, \ldots, a_{1n}], \ldots, [a_{m1}, \ldots, a_{mn}]\ ]$. Да се напише програма на Пролог, която:

а) по зададени представяния L_A и L_B на целочислените матрици A и B от една и съща размерност връща представянето на тяхната сума $C = A + B$;

б) по зададени представяне L_A на матрицата A и номер на стълб j, $1 \leq j \leq n$, връща списъка от елементите на j-ия стълб на A.

Задача 8. (10 точки) Дадено е пространство от състояния $V = \{v_0, v_1, v_2, \ldots, v_n\}$. Състоянието v_0 е начално, а v_n – крайно. Дадена е и функция на преходите $p(v)$, която за всяко състояние v определя множеството от състояния, които са негови непосредствени наследници. Цената на всеки преход между две състояния се определя от функцията $g(v_i, v_j)$. Да се реализира подходящ алгоритъм за намиране на път с минимална цена между началното и крайното състояние, използващ информирано търсене. За реализацията да се използват константно зададени в програмата входни данни. Да се опише алгоритъмът формално с псевдокод или като конкретна реализация на един от езиците C, C++, JAVA, Scheme или Prolog.

Задача 9. (10 точки)
Дадена е базата от данни Movies.



Таблицата ***Studio*** съдържа информация за филмови студия:
- *name* – име, първичен ключ;
- *address* – адрес.

Таблицата ***Movie*** съдържа информация за филми. Колоните *title* и *year* заедно формират първичния ключ.
- *title* – заглавие;
- *year* – година, в която филмът е заснет;
- *length* – дължина в минути;
- *incolor* – '**Y**' за цветен филм и '**N**' за черно-бял;
- *studioname* – име на студио, външен ключ.

Таблицата ***MovieStar*** съдържа информация за филмови звезди:
- *name* – име;
- *address* – адрес;
- *gender* – пол, '**M**' за мъж и '**F**' за жена;
- *birthdate* – рожденна дата.

Таблицата ***StarsIn*** съдържа информация за участието на филмовите звезди във филмите. Трите колони заедно формират първичния ключ. Колоните *movietitle* и *movieyear* образуват външен ключ.
- *movietitle* – заглавие на филма;
- *movieyear* – година на заснемане на филма;
- *starname* – име на филмовата звезда, външен ключ.

А) (4 точки) Да се посочи заявката, извеждаща всички актриси, които не са играли в нито един филм с име, започващо с буквата А. Актриси, за които в базата от данни няма информация за техните участия, също трябва да бъдат изведени.

```
А)
SELECT DISTINCT NAME
FROM MOVIESTAR
LEFT JOIN STARSIN
  ON NAME = STARNAME
WHERE GENDER = 'F'
  AND MOVIETITLE NOT LIKE 'A%';
```

```
Б)
SELECT NAME
FROM MOVIESTAR
JOIN STARSIN ON STARNAME = NAME
WHERE GENDER = 'F'
  AND MOVIETITLE LIKE 'A%'
GROUP BY NAME
HAVING COUNT(*) = 0;
```

```
В)
SELECT MS.NAME
FROM MOVIESTAR MS
WHERE MS.GENDER = 'F'
  AND NOT EXISTS
    (SELECT 1
     FROM STARSIN SI
     WHERE SI.STARNAME = MS.NAME
       AND SI.MOVIETITLE LIKE 'A%');
```

```
Г)
SELECT NAME
FROM MOVIESTAR
WHERE GENDER = 'F' AND NAME IN
  (SELECT DISTINCT STARNAME
   FROM STARSIN
   WHERE NOT MOVIETITLE LIKE 'A%');
```

Б) (6 точки) Да се посочи заявката, която за всяка филмова звезда (без значение от пола), родена преди 1990 г., извежда възрастта, на която е играла за първи път във филм. Звезди, за които няма информация за техните участия във филми, не трябва да бъдат извеждани.

```
А)
SELECT NAME, MIN( MOVIEYEAR - YEAR(BIRTHDATE) ) AS DEBUT_AGE
FROM MOVIESTAR
JOIN STARSIN ON NAME = STARNAME
WHERE YEAR(BIRTHDATE) < 1990
GROUP BY NAME;

Б)
SELECT NAME, MIN(MOVIEYEAR) - YEAR(BIRTHDATE) AS DEBUT_AGE
FROM MOVIESTAR
JOIN STARSIN ON NAME = STARNAME
WHERE YEAR(BIRTHDATE) < 1990
GROUP BY NAME;

В)
SELECT NAME, MIN(MOVIEYEAR) - YEAR(BIRTHDATE) AS DEBUT_AGE
FROM MOVIESTAR
LEFT JOIN STARSIN ON NAME = STARNAME AND YEAR(BIRTHDATE) < 1990
GROUP BY NAME
HAVING MIN(MOVIEYEAR);

Г)
SELECT DISTINCT MS.NAME, MOVIEYEAR - YEAR(BIRTHDATE) AS DEBUT_AGE
FROM MOVIESTAR MS, STARSIN
WHERE MS.NAME = STARNAME AND YEAR(BIRTHDATE) < 1990
HAVING MOVIEYEAR <= ALL (SELECT MOVIEYEAR
                         FROM STARSIN SI
                         WHERE SI.STARNAME = MS.NAME);
```