СОФИЙСКИ УНИВЕРСИТЕТ „СВ. КЛИМЕНТ ОХРИДСКИ“

ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА

ДЪРЖАВЕН ИЗПИТ

ЗА ПОЛУЧАВАНЕ НА ОКС “БАКАЛАВЪР ПО КОМПЮТЪРНИ НАУКИ”

ЧАСТ I (ПРАКТИЧЕСКИ ЗАДАЧИ)

11.9.2014 г.

Моля, не пишете в тази таблица!			
Зад. 1		Зад. 5	
Зад. 2		Зад. 6	
Зад. 3		Зад. 7	
Зад. 4		Зад. 8	
Крайна оценка:			

Драги абсолвенти:

- Попълнете факултетния си номер в горния десен ъгъл на всички листа;
- Пишете само на предоставените листове без да ги разкопчавате;
- Ако имате нужда от допълнителен лист, можете да поискате от квесторите;
- Допълнителните листа трябва да се номерират, като номерата продължават тези от настоящия комплект;
- Всеки от допълнителните листа трябва да се надпише най-отгоре с вашите три имена и факултетен номер.
- **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 решението на задачата не се побира в един лист, трябва да поискате нов бял лист от квесторите. В такъв случай отново трябва да започнете своето решение на листа с условието на задачата и в края му да напишете „Продължава на лист № X”, където X е номерът на допълнителния лист, на който е вашето решение.
- Черновите трябва да бъдат маркирани, като най-отгоре на листа напишете „ЧЕРНОВА“.
- На един лист не може да има едновременно и чернова и белова.
- Времето за работа по изпита е 3 часа

Изпитната комисия ви пожелава успешна работа!

Задача 1. (10 т.) Двуделен граф е неориентиран граф $G(V, E)$, такъв че съществува разбиване на V на непразни подмножества V_1, V_2, които наричаме дялове, такива че за всяко ребро (u, v) в графа е вярно, че u принадлежи на единия дял, а v принадлежи на другия дял. k-регулярен граф е неориентиран граф, в който всички върхове са от една и съща степен k.

а) Докажете, че във всеки двуделен граф, сумата от степените на върховете в единия дял е равна на сумата от степените на върховете в другия дял.

б) Докажете, че ако $G(V, E)$ е k-регулярен двуделен граф и $k > 0$, то двата дяла имат един и същи брой върхове.

Задача 2. (10 т.) Даден е детерминираният краен автомат

$$A = \langle \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}, \{a, b, c\}, q_0, \delta, \{q_0, q_3, q_4, q_5, q_7\} \rangle$$

с функция на преходите δ, определена както следва:

δ	a	b	c
q_0	q_7	q_0	q_6
q_1	q_2	q_2	q_5
q_2	q_1	q_6	q_0
q_3	q_5	q_3	q_7
q_4	q_7	q_5	q_4
q_5	q_3	q_5	q_6
q_6	q_6	q_1	q_4
q_7	q_0	q_7	q_3

Да се построи минимален детерминиран краен автомат A', еквивалентен на A.

Задача 3. (10 т.) В дясната страна на листа опишете какво очаквате да бъде изведено на стандартния изход (терминала), като резултат от изпълнението на следната програма на C, в която са използвани системни примитиви на ОС UNIX и LINUX:

```
main( )
{
   int a = 1000;

   if ( fork() )
   {
      a /= 2;
      printf ("\nValue of a = %d", a);
   }

   else
   {
      if ( fork() )
      {
         a*=2;
         printf ("\nValue of a = %d", a);

         if  ( execlp("ls","ls", "-l", 0 ) == -1 )
         {
            a = a + 2;
            printf ("\nValue of a = %d", a);
         }
      }
      else
      {
         a+=2;
         printf ("\nValue of a = %d", a);
      }
   }

   a++;
   printf ("\nValue of a = %d", a);
}
```

Задача 4. (10 т.) *Задачата да се реши на езика C++ или Java. В началото на вашето решение посочете кой език сте избрали.*

Дадени са координатите на N-точки, които са записани в масивите `float x[N], y[N]` по следния начин: координатите на `i`-тата точка са `(x[i], y[i])`.

Напишете функция `square`, която получава като аргументи броя на точките N и два масива `X` и `Y` съдържащи координатите им и извежда на екрана координатите на центъра и страната на най-малкия квадрат със страни успоредни на координатните оси, който обхваща всички дадени точки (всички дадени точки са във вътрешността му или на страните му).

Ако решавате задачата на Java, достатъчно е да напишете статична функция, която решава задачата.

Задача 5. (10 т.) *Задачата да се реши на езика C++ или Java. В началото на вашето решение посочете кой език сте избрали.*

Нека GameBoard е предварително дефинирана квадратна матрица от цели числа с размери N x N, представяща игрова дъска. Всеки елемент в матрицата има стойност 0 („земя”), 1 („огън”) или 2 („вода”). За две позиции в матрицата (i, j) и (i', j') казваме, че са съседни, ако $|i - i'| \leq 1$ и $|j - j'| \leq 1$.

А) Да се дефинира структура `Point`, описваща позиция на игровата дъска. Да се дефинира абстрактен клас (или интерфейс) `GamePlayer`, който описва играч на игровата дъска със следните операции:

- `getPosition`() – Връща позицията на играча на дъската;
- `allowedMoves`() – Връща списък (колекция) с всички възможни позиции, до които играчът може да достигне с един ход.

Б-1) Да се дефинира клас `Knight`, наследник на `GamePlayer`, описващ „сухопътен рицар“. Рицарят може да се придвижва само в такава съседна позиция, която е „земя” и не е в съседство с „огън”. Пример за достижими позиции за рицаря `K` е показан на диаграмата вдясно.

1	0	1
0	K	2
0	**0**	2

Б-2) Да се дефинира клас `SeaMonster`, наследник на `GamePlayer`, описващ „морско чудовище“. Морското чудовище може да се придвижва с произволен брой позиции по хоризонтала или по вертикала, но само по „вода”. Пример за достижими позиции за чудовището `S` е показан на диаграмата вдясно.

1	1	0	2	0
0	**2**	1	0	2
2	S	**2**	**2**	1
1	1	2	2	0
2	2	1	1	1

В) „Война“ наричаме такава подредба на играчите по дъската, при която на някоя от съседните позиции на всеки играч има друг играч. Да се дефинира функция

```
allMoves ([подходящ тип] players[, …])
```

която по даден списък (колекция) `players`, съдържащ произволен брой разнородни играчи, извежда на стандартния изход всеки възможен ход на играч от `players` такъв, че след изпълнението му списъкът с играчи да описва война. Информацията за ходовете да съдържа типа на играча, старата позиция и новата позиция.

Пример:

```
Knight (0,0) -> (1,1)
SeaMonster (2,2) -> (5,2)
```

Забележка: реализирайте всички конструктори и други операции, които смятате, че са необходими на съответните класове.

Задача 6. (10 т.) *Задачата да се реши на езика C++ или Java. В началото на вашето решение посочете кой език сте избрали.*

Двусвързан списък от цели числа се описва с двойка референции (указатели) съответно към началото и края на двусвързана верига от тройни клетки, представени по следния начин:

C++
```
struct Node {
  Node *next, *prev;
  int  data;
};
```

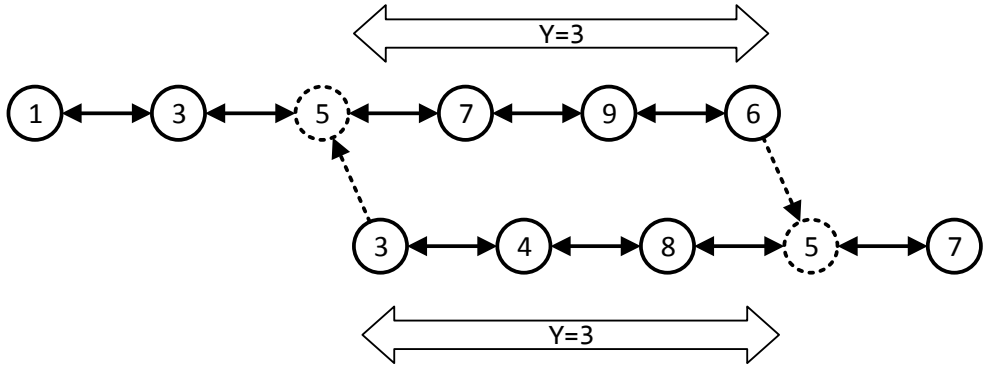
Java
```
public class Node {
  public Node next, prev;
  public int data;
}
```

Казваме, че двусвързаният списък L1 може да бъде „снаден“ с двусвързания списък L2 в числото M, ако има кутия A в L1 и кутия B в L2, такива че:

- `A.data` и `B.data` са равни на M;
- A е на разстояние Y от края на L1;
- B е на разстояние Y от началото на L2.

Разстояние между две кутии е броят на връзките, през които се преминава от едната до другата кутия. Снаждането на списъците се осъществява както е показано на диаграмата по-долу, като получената структура наричаме „снаден списък”.

Пример: Снаден списък, получен от снаждането на два списъка в числото 5 при Y = 3



А) Да се реализира функция `join`, която „снажда“ два двусвързани списъка L1 и L2, ако това е възможно. В случай, че снаждането може да се получи по няколко различни начина, да се избере този, при който разстоянието Y е минимално.

Б) Да се реализира булева функция `isJoined`, която по двойка референции (указатели) към начало и края на двусвързана верига проверява дали веригата е снаден списък. **Забележка:** да се счита, че подадената верига е двусвързан или снаден списък, т.е. не е нужно функцията да може да обработва друг вид вериги.

В) Да се реализира функция `sum`, която по даден снаден списък намира сумата на всичките му елементи.

Забележка: Ако решавате задачата на езика Java, за всяко подусловие е достатъчно да реализирате статична функция, която получава вход съгласно спецификацията и извършва нужните действия.

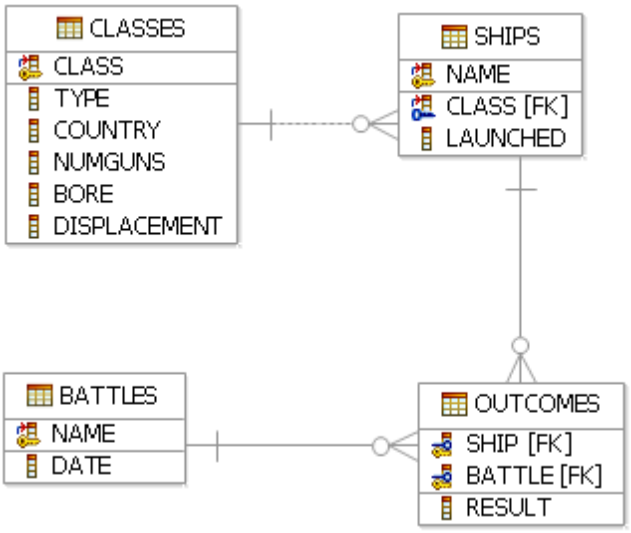
Задача 7. (10 т.) *Задачата да се реши на езика Scheme или Haskell. В началото на вашето решение посочете кой език сте избрали*.

Нека е даден списък L, който може да съдържа елементи от произволен тип. Напишете функция `permutations`, която получава такъв списък и връща списък с всички пермутации (възможни пренареждания) на неговите елементи. Резултатът да се върне като списък от списъци, в който всеки подсписък представя една пермутация на елементите на L.

Пример (Scheme):
`(permutations '(1 2 3))` → `( (1 2 3) (1 3 2) (2 1 3) (2 3 1) (3 1 2) (3 2 1) )`

Пример (Haskell):
`permutations [1,2,3]` → `[ [1,2,3], [1,3,2], [2,1,3], [2,3,1], [3,1,2], [3,2,1] ]`

Задача 8. (10 т.) Дадена е базата от данни Ships, в която се съхранява информация за кораби (*Ships*) и тяхното участие в битки (*Battles*) по време на Втората световна война. Всеки кораб е построен по определен стереотип, определящ класа на кораба (*Classes*).



Таблицата ***Classes*** съдържа информация за класовете кораби:

- *class* – име на класа, първичен ключ;
- *type* – тип (‘bb’ за бойни кораби и ‘bc’ за бойни крайцери);
- *country* – държавата, която строи такива кораби;
- *numGuns* – броят на основните оръдия;
- *bore* – калибърът им (диаметърът на отвора на оръдието в инчове);
- *displacement* – водоизместимост (в тонове).

Таблицата ***Ships*** съдържа информация за корабите:

- *name* – име на кораб, първичен ключ;
- *class* – име на неговия клас, външен ключ към Classes.class;
- *launched* – годината, в която корабът е пуснат на вода.

Таблицата ***Battles*** съхранява информация за битките:

- *name* – име на битката, първичен ключ;
- *date* – дата на провеждане.

Таблицата ***Outcomes*** съдържа информация за резултата от участието на даден кораб в дадена битка (колоните ship и battle заедно формират първичния ключ):

- *ship* – име на кораба, външен ключ към Ships.name;
- *battle* – име на битката, външен ключ към Battle.name;
- *result* – резултат (потънал-‘sunk’, повреден – ‘damaged’, победил – ‘ok’).

За така описаната база данни, решете следните задачи:

1. Оградете буквата на заявката, която извежда имената на всички кораби, пуснати на вода в година, в която е имало битка (не е задължително корабът да е участвал в нея).

А)
```
select name
from ships
where launched = any (
select year(date)
from battles
where count(*) >= 1);
```

Б)
```
select distinct ships.name
from battles , ships
where launched = year(date);
```

В)
```
select name
from battles
where exists (select distinct *
from ships
where year(date) = launched);
```

Г)
```
select distinct name
from ships
join battles
on launched = year(date);
```

2. Оградете буквата на заявката, която за всички държави, които имат най-много 3 (евентуално 0) кораба, извежда името на държавата и броя потънали кораби (който също може да бъде 0).

А)
```
select country, count(result)
from classes c
left join ships s
on c.class = s.class
left join outcomes o
on s.name = o.ship
where o.result = 'sunk'
group by country
having count(ship) <= 3;
```

Б)
```
select country,
count(result is 'sunk')
from ships, classes, outcomes
where count(ship) <= 3
or ship is null;
```

В)
```
select distinct classes.country,
sunk_cnt
from classes
right join (select country,
count(*) as sunk_cnt
 from classes c
 join ships s
 on c.class = s.class
 join outcomes o
 on s.name = o.ship
 where result = 'sunk'
 group by country) sunk
on classes.country = sunk.country
where sunk_cnt <= 3;
```

Г)
```
select country,
count(result = 'sunk') as sunk_cnt
from ships s
join outcomes o on s.name = o.ship
right join classes c
on s.class = c.class
where count(*) <= 3
group by country, sunk_cnt;
```

Д)
```
select distinct country,
(select count(*)
 from classes c2 join ships s on c2.class = s.class
 join outcomes o on s.name = o.ship
 where c2.country = c.country and result = 'sunk')
from classes c
where (select count(*)
       from classes c2 join ships s  on c2.class = s.class
       where c2.country = c.country) <= 3;
```

Ч Е Р Н О В А

Ч Е Р Н О В А

Ч Е Р Н О В А