СОФИЙСКИ УНИВЕРСИТЕТ „СВ. КЛИМЕНТ ОХРИДСКИ“

ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА

ДЪРЖАВЕН ИЗПИТ

ЗА ПОЛУЧАВАНЕ НА ОКС “БАКАЛАВЪР ПО СОФТУЕРНО ИНЖЕНЕРСТВО”

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.) 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 2. (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 3. (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 4. (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.

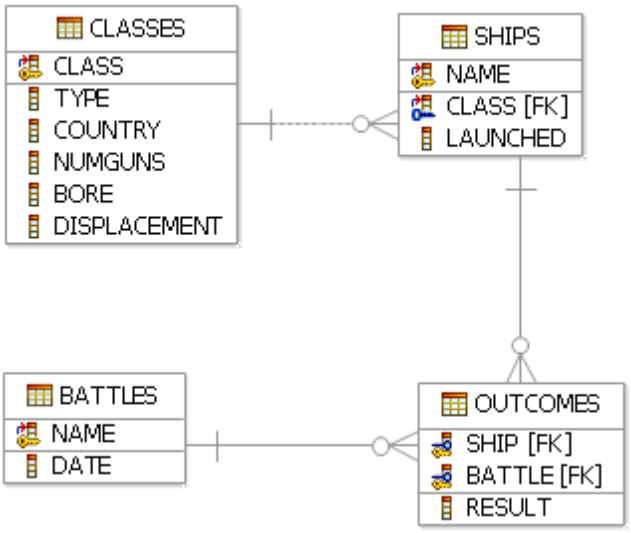
Задача 5. (10 т.) Да се проектира софтуерната архитектура на система, според следните изисквания, предназначена за подпомагане на бегачи-аматьори (бягане за здраве) и която е съставена от две части – полева (която работи докато човек бяга) и онлайн част. Към системата има следните изисквания:

- Полевата част измерва био-показатели на бегача (кръвно налягане, пулс, температура) и изминатото разстояние (чрез GPS).
- Всички измерени от полевата част на системата данни, се изпращат към онлайн частта.
- След обработка в онлайн частта към бегача се връща обратна връзка и съответните препоръки – за промяна на маршрута, за увеличаване/намаляване на скоростта на бягането, за почивка или за спиране на тренировката за деня.
- Обратната връзка става по избран от бегача начин: на дисплея или през слушалките на мобилно устройство.
- Обратната връзка трябва да достига до бегача не по-късно от 4 секунди след измерване на показателите от полевата част.
- Всеки бегач сам създава през WWW профил в онлайн частта на системата.
- В профила се съхраняват всички измерени от полевата част на системата данни за съответния бегач.
- Всеки бегач може да добавя, като приятели, профилите на други бегачи (аналогично на социалните мрежи).

Задача 6. (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;
```

Задача 7. (10 т.) Дадена е информационна система, която съхранява информация за Обиколката на Франция (le Tour de France) през 2014 г. Базата от данни трябва да съдържа следната информация:

Отбори (Teams)

- Име на отбор (tname) – низ до 20 символа, първичен ключ
- Държава (tcountry), за която се състезава отбора, низ точно 3 символа
- Брой победи на предходни състезания le Tour de France (num_tf), цяло **положително число**, може и NULL
- Брой етапни победи на предходни състезания le Tour de France (num_stf), цяло **положително число**, може и NULL
- Брой спечелени жълти фланелки на предходни състезания le Tour de France (num_yj), цяло **положително число**, може и NULL

Колоездачи (Riders)

- Име на колоездач (rname) – низ до 50 символа, първичен ключ
- Номер на фланелка (rnum), цяло **положително число**
- Дата на раждане (birthdate) - дата
- Височина на колоездач (height) - цяло **положително число**
- Килограми на колоездач (weight) - реално **положително число**
- Държава (rcountry) от която е колоездача - низ точно 3 символа
- Град (rcity) от който е колоездача - низ до 20 символа
- Име на отбор (tname) за който се състезава колоездача - низ до 20 символа, външен ключ към отбор на колоната Име на отбор от таблицата Отбори

Етапи (Stages)

- Номер на етап (snumber) – цяло **положително число**, първичен ключ
- Дата на провеждане на етапа (sdate) – дата
- Километри на етапа (km) - реално **положително число**
- Град начало на етапа (scity) - низ до 30 символа
- Град край на етапа (ecity) - низ до 30 символа

Обиколка (Tour)

- Идентификационен номер (id) – цяло положително число, първичен ключ
- Номер на етап (snumber) – цяло **положително число**, външен ключ към колоната Номер на етап от таблицата Етапи
- Име на колоездач (rname) – низ до 50 символа, външен ключ към колоната Име на колоездач от таблицата Колоездачи
- Мястото на което се е класирал колоездача (place) - цяло **положително число**
- Спечелени точки от етапа (points) - цяло число
- Време за което е завършил етапа (ttime) - от тип време
- Дали е спечелил бяла фланелка (white) - цяло число може да бъде 0 или 1
- Дали е спечелил жълта фланелка (yellow) - цяло число може да бъде 0 или 1
- Дали е спечелил зелена фланелка (green) – цяло число може да бъде 0 или 1

Като използвате SQL създайте така описаните таблици и ограничения.

Задача 8. (10 т.) Да се направи информационен модел на електронна автобиография на хора, който да замести хартиения документ. Да се аргументира изборът за всеки обект от модела, структурата на модела и броя на появяванията на обектите. Автобиографията съдържа: информация за имената, дата на раждане, месторождение и гражданство, информация за адреса (държава, град, улица, номер на улица), телефон и email, информация за образователните институции, които е завършил, специалностите, по които се е обучавал, вида образование, което е завършил и година на дипломиране. Това съдържание е записано в XML документ, в който има информация за нула, един или повече души. Да се състави описание на документния му тип с използване на DTD или XML Schema. Да се създадат примерен XML документ, в който има данни за поне двама души, и съответен DTD или XML Schema документ

ЧЕРНОВА

Ч Е Р Н О В А

Ч Е Р Н О В А