СОФИЙСКИ УНИВЕРСИТЕТ
“СВ. КЛИМЕНТ ОХРИДСКИ”

ФАКУЛТЕТ ПО МАТЕМАТИКА
И ИНФОРМАТИКА

ДЪРЖАВЕН ИЗПИТ
ЗА ПОЛУЧАВАНЕ НА ОКС “БАКАЛАВЪР ПО ИНФОРМАТИКА”

ЧАСТ I (ПРАКТИЧЕСКИ ЗАДАЧИ)
05.07.2008 г.

Време за работа – 3 часа

Драги абсолвенти:
- *Попълнете факултетния си номер на всички страници;*
- *За всяка от задачите, беловата с решението може да е само на листите, на които е изписано условието на съответната задача.*

Изпитната комисия ви пожелава успешна работа.

Задача 1. (10т.) Да се построи краен детерминиран автомат, еквивалентен на дадения краен недетерминиран автомат

$A = <\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}, \{a,b,c\}, q_0, \delta, \{q_0\}>$

с функция на преходите δ, дефинирана от следната таблица:

q	a	b	c
q_0	$\{q_0, q_1\}$	$\varnothing$	$\{q_3\}$
q_1	$\varnothing$	$\{q_2\}$	$\varnothing$
q_2	$\{q_0\}$	$\varnothing$	$\{q_2\}$
q_3	$\varnothing$	$\{q_5\}$	$\{q_4\}$
q_4	$\varnothing$	$\{q_5, q_6\}$	$\{q_2\}$
q_5	$\varnothing$	$\varnothing$	$\{q_4\}$
q_6	$\varnothing$	$\{q_0\}$	$\varnothing$

Задача 2. (13 т.) В масив `a[N,2]` от тип `double`, всеки ред съдържа 2 реални числа, които са координати на точка. Всички точки лежат на една окръжност (точките не са подредени в определена последователност).

1) Напишете програма (`C, C++` или `Java`), която:
   a) определя центъра и радиуса на окръжността;
   b) създава двумерен масив от `MxM` пиксела (модел на растерен екран - 1 байт на пиксел) такъв, че центърът му да отговаря на центъра на окръжността, а M се избира така, че да няма съседни последователни точки на разстояние по-малко от 5 пиксела;
   c) изчертава окръжността, като извиква предварително реализиран модул за растерно изчертаване на окръжност, работещ върху горния масив.

В рамките на 20 думи да се обоснове избора на модул за изчертаване - чрез алгоритъма на Брезенхам или чрез алгоритъма на средната точка.

Задача 3. (7 т.) Зададен е следния фрагмент от програма:

```
#define LST “ls”
main()
{
   int  pid, k=5, status;
   printf( “ Stoinostta na k = %d;”, k-2 );
   ++k;
   printf( “ Stoinostta na k = %d;”, k );
   execlp(LST,LST,0);
   if ( (pid  = fork() ) = = 0 ) k++;
   else  { wait( &status); --k ; }
   printf( “ Stoinostta na k = %d;”, k );
}
```

Като резултат от изпълнението на последователността от зададените оператори на стандартния изход ще се изведе:

Задача 4. (14 т.) Да се реализира подходящо представяне на двоично дърво за търсене. За избраното представяне да се реализират операциите *добавяне на елемент* и *търсене на елемент по стойност*. Да се състави функция, която по дадено такова дърво извлича в нарастващ ред всички елементи, които са на определена дълбочина.

Задача 5. (15 т.) Магазин продава два типа артикули – **бройни** и **количествени**. Една покупка на броен артикул се задава чрез името му, единичната цена и броя закупени артикули, който е цяло число, а една покупка на количествен – чрез името му, цената за килограм и закупеното количество – реално число, което показва колко килограма са закупени.

а) Да се реализира абстрактен базов клас, който обединява общата функционалност за двата вида покупки и има виртуален метод за пресмятане на стойност на една покупка. Да се реализират два производни класа, представящи покупките на двата типа артикули.

б) Да се реализира клас ***касова_бележка***. Бележката трябва да има номер и масив от покупки. В класа за касова бележка да има метод, който пресмята общата стойност на всички покупки от масива.

в) Демонстрирайте използването на класовете в подходяща кратка програма.

ЧЕРНОВА за задачи 4 и 5

Задача 6. (11 т.) Попълнете в празните полета текста, който се отпечатва на стандартния изход, в резултат на изпълнението на следните програмни конструкции:

```
#include <iostream>
using namespace std;
class Base
{ public:
     Base () {cout << "Base::Base()\n";}
     Base (const Base&) {cout << "Base::Base(Base&)\n";}
     virtual void f () {cout << "Base::f()\n";}
     void g () {cout << "Base::g()\n";}
};
class Derived : public Base
{ public:
     Derived () {cout << "Derived::Derived()\n";}
     Derived (const Derived&) {cout <<
                                "Derived::Derived(Derived&)\n";}
     void f () {cout << "Derived::f()\n";}
     void g () {cout << "Derived::g()\n";}
};
void fCopy (Base obj)
{    obj.f (); obj.g (); }
void fReference (Base *obj)
{    obj->f (); obj->g (); }

void main ()
{    Derived obj;
```

а)

```
     Base *pBase = &obj;
     Derived *pDerived = &obj;
     pBase->f ();
```

б)

```
     pBase->g ();
```

в)

```
     pDerived->f ();
```

г)

```
     pDerived->g ();
```

д)

```
     fCopy (obj);
```

е)

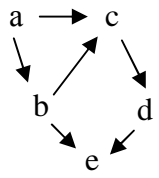
```
     fReference (&obj);
```

ж)

```
}
```

Задача 7. (13 т.) Да се дефинира функция ***(ways g u v)*** на езика Scheme, намираща броя на различните пътища между върха ***u*** и върха ***v*** в ориентирания ацикличен граф ***g***, представен чрез асоциативен списък на наследниците.

Пример. Ако е даден графът



Представен чрез списъка *g = '((a b c) (b c e) (c d) (d e) (e))*, то оценката на *(ways g ‘a ‘e)* е 3.

Задача 8. (11 т.) Да наречем редица на Трибоначи редицата дефинирана така: $a_1 = a_2 = a_3 = 1$, $a_{n+3} = a_{n+2} + a_{n+1} + 2a_n$. Без използването на вградени предикати с изключение на `=`, `is`, `<`, `>`, `=<`, `>=` и `not` да се дефинира на Пролог предикат `tribo(N,A)`, който по дадено естествено число `N` намира в `A` *N*-тия член на редицата на Трибоначи.

Задача 9. (6 т.) В базата данни със схема:

Classes(**class**, **type**, **country**, **numGuns**, **bore**, **displacement**)
Ships(**name**, **class**, **launched**)
Battles(**name**, **date**)
Outcomes(**ship**, **battle**, **result**)

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). Обикновено класът носи името на първия построен кораб от този клас.

Таблицата Classes съдържа информация за класовете кораби:

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

Таблицата Ships съдържа информация за корабите:

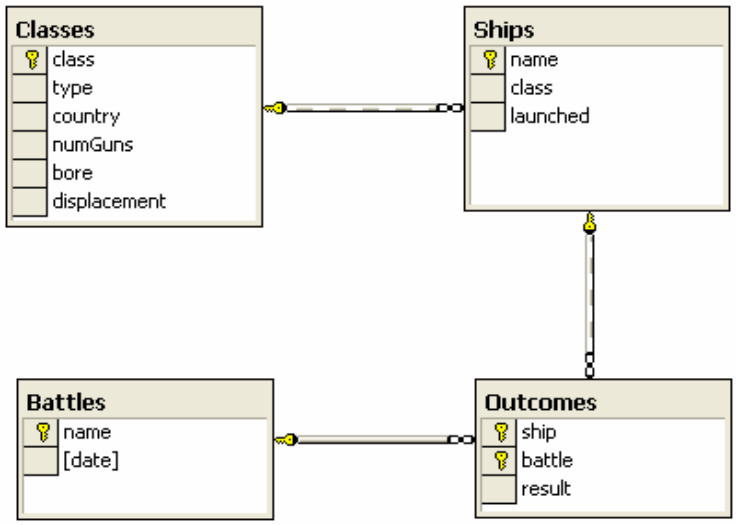
- **name – името на кораб, първичен ключ;**
- **class – името на неговия клас;**
- **launched – годината, в която корабът е пуснат на вода.**

Таблицата Battles съхранява информация за битките:

- **name – името на битката, първичен ключ;**
- **date – датата на провеждане.**

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

- **ship – името на кораба;**
- **battle – името на битката;**
- **result – резултатът (потънал – ‘sunk’, повреден – ‘damaged’, победил – ‘ok’).**



Посочете заявката, която извежда имената на битките, в които няма оцелели кораби (т.е. всички участвали кораби са потънали):

а)
```
select distinct b.name
from Battles b
join Outcomes o on b.name = o.battle
where o.result = 'sunk';
```
б)
```
select o.battle
from Outcomes o
group by o.battle,o.result
having o.result = 'sunk';
```
в)
```
select distinct o.battle
from Outcomes o
left join Outcomes o1 on o.battle  = o1.battle and
                         o.ship    = o1.ship and
                         o1.result = 'sunk'
group by o.battle
having count(o.ship)= count(o1.ship);
```
г)
```
select o.battle
from Outcomes o
group by o.battle
having count(case when o.result = 'sunk' then 'Y' end)=0;
```