СОФИЙСКИ УНИВЕРСИТЕТ "СВ. КЛИМЕНТ ОХРИДСКИ"
ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА

ДЪРЖАВЕН ИЗПИТ
ЗА ПОЛУЧАВАНЕ НА ОКС "БАКАЛАВЪР ПО ИНФОРМАТИКА"
17-18.03.2007 г.

ЧАСТ I (ПРАКТИЧЕСКИ ЗАДАЧИ)

Задача 1 (2 т.)
Обяснете предназначението на следната функция :

```
int m (int a, int b) {
        int M[2]={0};
        return( M[a>b]=a, M[b>a]=b, M[0] ); }
```

Задача 2 (общо 10 т.)
При условията на следната дефиниция:

```
bool check (int A[ ], int n) {
        int N[3]={0};
        for ( int j=n; --j; )
           (N[1-(A[j-1]<A[j])+(A[j-1]>A[j])])++;
        return N[0]== n-1; }
```

a) **(3 т.)** Обяснете накратко какво свойство проверява така дефинираната функция.

b) **(1 т.)** Коя от двете алгоритмични идеи: "∀" или "∃" е използвана при проверката? (заградете с кръгче)

c) **(6 т.)** Напишете нов вариант на същата проверка, но като приложите другата алгоритмична идея:

```
bool check (int A[ ], int n)
```

Задача 3 (3 т.)
При условията на следните дефиниции:

```
char S[20] = "--12345---" ;
char *f(char *p, char *q) {
        if ( *p == '-' ) p = p + 1 ;
        if ( *q == '-' ) { *q = 0 ; q = q - 1 ; }
        if ( *p == '-' || *q == '-' )
                return f(p, q) ;
        return p ; }
```

напишете какво ще се получи при изпълнението на оператора:

```
cout << '(' << f( S, S+strlen(S)-1 ) << "),(" << S << ")\n" ;
```

Задача 4 (6 т.)

Отбележете и обяснете грешките в програмата. Поправете ги така, че да се получи работеща програма. Намерете резултата от изпълнението й.

```
#include <iostream.h>
class A
  {public:
            A(int, int = 1);
            void print();
            int f_Ax() const;
            int f_Ay() const;
    private:
             int x, y;
  };
  A::A(int a, int b)
  {x = a;
   y = b;
  }
  void A::print()
  {cout << x << " " << y << endl;
  }
  int A::f_Ax() const
  {return x;
  }
  int A::f_Ay() const
  {return y;
  }
  class B
  {public:
            B(double, A);
            void print() const;
            double f_Bx();
            A f_Ba() const;
   private:
            double x;
            A a;
  };
  B::B(double d, A e)
  {x = d;
   a = e;
  }
  void B::print() const
  {cout << x << endl;
   a.print();
  }
  double B::f_Bx() const
  {cout << a.f_Ax() << " " << a.f_Ay() << endl;
   return x;
  }
  A B::f_Ba()
  {return a;
  }

  void main()
  {A a(13), x;
    a.print();
    cout << x.x << " " << x.y << endl;
    B b(6.5, a);
    b.print();
  }
```

Задача 5 (10 т.)

Разгледайте програмата:

```
#include <iostream.h>
class Base
{public:
   virtual void virt1()
   {cout << "Base::virt1() \n";
   }
   Base()
   {cout << "Base()\n";
     virt1();
     virt2();
     virt3();
    }
  private:
      virtual void virt2()
      {cout << "Base::virt2()\n";
       }
  protected:
       virtual void virt3()
      {cout << "Base::virt3()\n";
      }
};
class Der1 : public Base
{ void virt1()
    {cout << "Der1::virt1()\n";
     }
  protected:
     void virt2()
     {cout << "Der1::virt2()\n";
      }
  public:
      void virt3()
      {cout << "Der1::virt3()\n";
      }
};
class Der2 : public Der1
{protected:
   void virt1()
    {cout << "Der2::virt1()\n";
     }
  public:
    void virt2()
    {cout << "Der2::virt2()\n";
    }
  private:
    void virt3()
    {cout << "Der2-virt3()\n";
    }
};
void main()
{ Base b;
   Der1 d1; Der2 d2;
   Base *p = &d1;
   Der1 *q = &d2;
   b.virt1();
   b.Base();
   p->virt1();
   p->virt2();
   p->virt3();
   q->virt1();
   q->virt2();
   q->virt3();
```

```
    q->Base();
    p = &d2;
    p->virt1();
    p->virt2();
    p->virt3();
    Der1 *r = new Der2;
    r->virt2();
    r->virt1();
    r->virt3();
   delete r;
  }
```

a) Намерете и обяснете грешките в процедурата main на горната програма.
b) Кои връзки в програмата се разрешават статично и кои - динамично?
c) Какъв е резултатът от изпълнението на програмата след отстраняване на неправилните обръщения към виртуалните функции?

Задача 6 (общо 16 т.)

Шаблонът на класа queue реализира опашка с върхове от тип T.

```
template <class T>
struct elem_q
{T inf;
 elem_q<T>* link;
};
template <class T>
class queue
{public:
  // канонично представяне
  queue();
  ~queue();
  queue(queue const &);
  queue& operator=(queue const &);
  // InsertElem включва x в неявната опашка
  void InsertElem(T const& x);
   // DeleteElem изключва елемент от неявната опашка и го записва в x
  int DeleteElem(T & x);
  void print();                  // извежда на екрана неявната опашка
  bool empty() const;            // проверява дали е празна неявната опашка
 private:
  elem_q<T> *front, *rear;
  void delqueue();
  void copy(queue const&);
};
```

a) **(2 т.)** Реализирайте член-функциите от каноничното представяне на шаблона на класа.

b) **(5 т.)** Предефинирайте оператора < като член-функция на шаблона на класа queue<T>, така че да установява дали опашка се съдържа в друга опашка.
(Опашката от цели числа 1, 2, 3 се съдържа в опашката 5, 2, 3, 1, 2, 3, 4 и не се съдържа в опашката 1, 2, 4, 3, 5, 6).

c) **(5 т.)** Дефинирайте външна функция, която слива две сортирани във възходящ ред опашки от числа.

d) **(4 т.)** Дефинирайте шаблон на външна функция с параметър T, която прилага функцията f: T → T над всеки от елементите на опашка с елементи от тип T.

Задача 7 (общо 12 т.)

Шаблонът на класа BinOrdTree реализира двоично-наредено дърво (ДНД) с върхове от тип T.

```
template <class T>
struct node_bin
{T inf;
 node_bin<T> *Left;
 node_bin<T> *Right;
};
template <class T>
class BinOrdTree
{public:
   BinOrdTree();
   ~BinOrdTree();
   BinOrdTree(BinOrdTree<T> const&);
   BinOrdTree<T>& operator=(BinOrdTree<T> const&);
   bool empty() const;              // проверява дали неявното ДНД е празно
   T RootTree() const;              // връща корена на неявното ДНД
   BinOrdTree LeftTree() const;          // връща лявото поддърво на неявното ДНД
   BinOrdTree RightTree()const; // връща дясното поддърво на неявното ДНД
   void DeleteNode(T const&);           // изтрива указания елемент от неявното ДНД
  // Create3 създава ДНД по указани корен, ляво и дясно двоично-наредени поддървета
   void Create3(T const&,BinOrdTree<T> const&, BinOrdTree<T> const&);
   // …
  private:
   node_bin<T> *root;
   // DeleteTree изтрива двоично-нареденото дърво, зададено чрез указателя p
   void DeleteTree(node_bin<T>* &p) const;
   // CopyTree копира указаното в неявното двоично-наредено дърво
   void CopyTree(BinOrdTree<T> const&);
   void Copy(node_bin<T> * &, node_bin<T>* const&) const;
  // MinTree намира минималния елемент x на непразното неявно ДНД и двоично-
  // наредено дърво mint, получено след изключване на x от неявното ДНД
   void MinTree(T& x, BinOrdTree<T>& mint) const;
 // ……
};
```

a) **(2 т.)** Дефинирайте следните член-функции на шаблона:
   BinOrdTree();
   BinOrdTree(BinOrdTree<T> const&);

b) **(5 т.)** Член-функцията void DeleteNode(T const&); на шаблона на класа BinOrdTree изключва указания като параметър елемент от неявното двоично-наредено дърво. Релизирайте я.

c) **(2 т.)** Предефинирайте оператора ==, така че да установява дали две двоично-наредени дървета са равни, чрез шаблон на функция-приятел на шаблона на класа BinOrdTree<T>.

d) **(3 т.)** Проверете дали дадено двоично-наредено дърво с елементи от тип T е балансирано (за всеки негов връх дълбочините на лявото и дясното му поддървета се различават най-много с 1).

Задача 8 (5 т.)

Да се напише командна процедура с един параметър - идентификатор на потребител. Процедурата прочита от стандартния вход цяло положително число и проверява дали в сесия работи потребител с подадения като позиционен параметър идентификатор и дали броят на сесиите на този потребител е по-голям от прочетеното число. Ако да - процедурата изпраща на същия потребител съобщение със следния текст: ”Моля, свържете се незабавно със системния администратор”.

Задача 9 (3 т.)

Даден е следният фрагмент от програма:

```
#define LST “date”
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 );
```

Напишете вдясно какво ще бъде изведено на стандартния изход като резултат от изпълнението на този фрагмент.

Задача 10 (8 т.)

Да се определи за какви стойности на **n≥2** е шеферова двоичната функция
$\mathbf{f(x_1,\ldots,x_n)=1\oplus x_1x_2\oplus x_2x_3\oplus\ldots\oplus x_ix_{i+1}\oplus\ldots\oplus x_{n-1}x_n}$.

Задача 11 (6 т.)

Да се построи краен автомат, разпознаващ езика, представен чрез регулярния израз **(ab*+ac)b+(a+b)(cca*b)***.

Задача 12 (2 т.)

Кои от изброените формули са логически следствия от формулата $(p \Rightarrow q) \& p$:

a) p
b) q
c) $\neg q$
d) $p \Leftrightarrow q$

Задача 13 (3 т.)

Дадена е следната програма на Prolog:

female(mary).
driver(jane).

На кои от изброените цели Prolog ще отговори с “no”:

a) not(female(X)), driver(X).
b) driver(X), not(female(X)).
c) not(female(X), driver(X)).
d) not(female(jane)).

Задача 14 (4 т.)

Нека F е съвкупността от всички едноместни частични функции в множеството N на естествените числа. Кои от изброените оператори

$\Gamma\_i: F \longrightarrow F$, $1 \le i \le 4$, НЕ са монотонни? Обосновете отговорите си.

a) $\Gamma\_1(f)(x) \cong$ if $x = 0$ then 1 else $x - 1$
b) $\Gamma\_2(f)(x) \cong$ if $x \equiv 0 \pmod 2$ then 1 else 0
c) $\Gamma\_4(f)(x) \cong$ if f е безкрайна then 1 else недефинирано
d) $\Gamma\_3(f)(x) \cong$ if f е безкрайна then недефинирано else 1

Задача 15 (5 т.)

Нека R е следната рекурсивна програма над типа данни Nat:

R: G(X, Y) where
F(X) = if $X \le 1$ then X else F(X–2)
G(X, Y) = if $X = 0$ then Y else G(X – 1, F(Y)).

Кои от изброените условия са верни за D_V(R) – денотационната семантика на R с предаване на параметрите по стойност:

a) $\forall x \forall y$ (!D_V(R)(x, y) $\Rightarrow$ D_V(R)(x, y) ≤ 1)
b) $\forall x \forall y$ ($x > 0 \Rightarrow \neg$!D_V(R)(x, y))
c) $\forall x \forall y$ ($\neg$!D_V(R)(x, y) $\Rightarrow x > 0$)
d) $\forall x \forall y$ (!D_V(R)(x, y) $\Rightarrow$ D_V(R)(x, y) $\equiv y \pmod 2$)

Задача 16 (3 т.)
Дефинирайте функция на езика Scheme, която връща като резултат броя на целите числа в интервала ***[a,b]*** (***a*** и ***b*** са две дадени цели числа, ***a≤b***), в десетичния запис на които се съдържа цифрата ***k***, като за целта реализира итеративен изчислителен процес.

Задача 17 (3 т.)
Дефинирайте функция на езика Scheme, която за даден списък ***l***, елементите на който са непразни списъци от числа, връща като резултат списък от поредните номера на тези елементи на ***l***, произведението на чиито елементи е положително число.

Задача 18 (2 т.)
Даден е ориентиран граф, представен чрез поредица от факти на Prolog от вида arc(<Node1>,<Node2>), всеки от които означава, че в графа съществува дъга с начало <Node1> и край <Node2>:

```
arc(s,a).   arc(s,b).   arc(s,c).   arc(a,d).   arc(b,d).   arc(b,e).
arc(c,d).   arc(c,f).   arc(d,g).   arc(e,g).   arc(f,g).
```

Дадена е също така поредица от факти на Prolog от вида h(<Node>,<Cost>), дефиниращи евристичната функция, с помощта на която се пресмята приближена стойност <Cost> на разстоянието от върха <Node> до върха “g”:

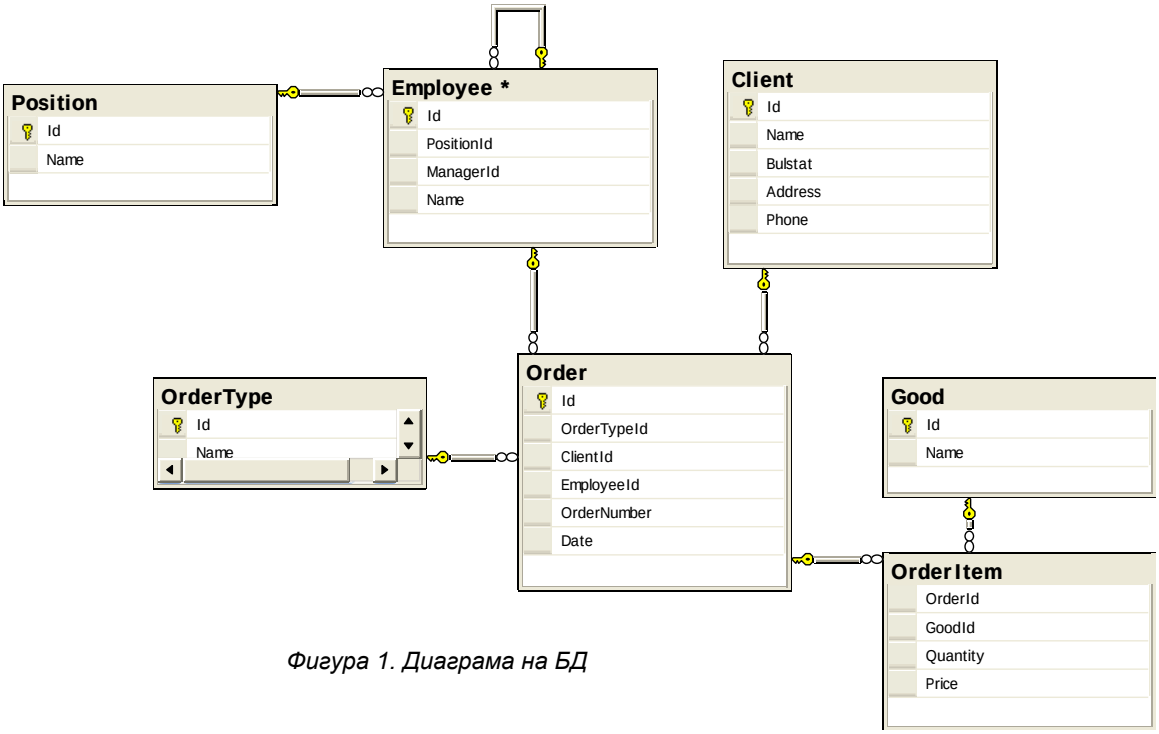
```
h(a,3).   h(b,2).   h(c,1).   h(d,3).
h(e,2).   h(f,3).   h(g,0).   h(s,5).
```

Ако задачата е да се намери път от върха “s” до върха “g” по метода Best-first Search, какво ще бъде полученото решение? Запишете решението като списък от съставящите пътя върхове на графа.

Задача 19 (3 т.)
Представете чрез разделена семантична мрежа (partitioned semantic net) следното изречение:
“Every player kicked a ball.”

Задача 20 (5 т.)

За управление на фактурите в търговска фирама е създадна база от данни. Диаграмата на БД е показана на фиг. 1.



Фигура 1. Диаграма на БД

Релационната схема на БД е:

Position(***Id***, Name)
OrderType(***Id***, Name)
Good(***Id***, Name)
Employee(***Id***, PositionId, ManagerId, Name)
Client(***Id***, Name, Bulstat, Address, Phone)
Order(***Id***, OrderTypeId, ClientId, EmployeeId, OrderNumber, Date)
OrderItem(***OrderId***, ***GoodId***, Quantity, Price)

където:

* **Table**				
Column	***Comment***	***Datatype***	***Null***	***Constraints***
* **Position**	Съхранява информация за позициите на служителите във фирмата			
Id	Генерира се автоматично	int	NOT NULL	Primary Key
Name	Име на позицията	varchar	NOT NULL	
* **OrderType**	Вид фактура			
Id	Генерира се автоматично	int	NOT NULL	Primary Key
Name	Вид на фактура. Възможни стойности са: - sell - buy	varchar	NOT NULL	
* **Good**	Стоки			
Id	Генерира се автоматично	int	NOT NULL	Primary Key
Name	Име на стоката	varchar	NOT NULL	
* **Employee**	Служители на фирмата			
Id	Генерира се автоматично	int	NOT NULL	Primary Key
PositionId	Длъжност на служителя - **Position.Id**	int	NOT NULL	Foreign Key
ManagerId	На кого е подчинен - **Employee.Id**	int	NULL	Foreign Key
Name	Име на стоката	varchar	NOT NULL	
* **Client**	Клиенти на фирмата			

Id	Генерира се автоматично	int	NOT NULL	Primary Key
Name	Име на клиент	varchar	NOT NULL	
Bulstat	Уникален идентификационен код (БУЛСТАТ/ЕГН)	varchar	NOT NULL	
Address	Адрес	varchar	NULL	
Phone	Телефон	varchar	NULL	
*** Order**	Фактури			
Id	Генерира се автоматично	int	NOT NULL	Primary Key
OrderTypeId	Вид на фактурата (покупка или продажба) - **OrderType.Id**	int	NOT NULL	Foreign Key
ClientId	Име на клиент - **Client.Id**	int	NOT NULL	Foreign Key
EmployeeId	Служител, който е въвел данните за фактура - **Employee.Id**	int	NOT NULL	Foreign Key
OrderNumber	Номер на фактура	varchar	NOT NULL	
Date	Дата на фактура	datetime	NOT NULL	
*** OrderItem**	Фактури			
OrderId	Фактура - **Order.Id**	int	NOT NULL	Primary Key, Foreign Key
GoodId	Стока - **Good.Id**	int	NOT NULL	Primary Key, Foreign Key
Quantity	Количество	decimal(18, 4)	NOT NULL	Foreign Key
Price	Единична цена	decimal(18, 4)	NOT NULL	Foreign Key

За всеки клиент, реализирал положителен оборот, да се изведе справка за оборота, който този клиент е направил за фирмата. Оборотът се получава, като от сумата на всички покупки (sell) се извади сумата на всички продажби (buy). Справката да има следния вид:

ClientName	Turnover