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

ДЪРЖАВЕН ИЗПИТ
ЗА ПОЛУЧАВАНЕ НА ОКС "БАКАЛАВЪР ПО ИНФОРМАТИКА"
16-17.09.2006 г.

ЧАСТ I (ПРАКТИЧЕСКИ ЗАДАЧИ)

Задача 1 (3 т.)
Да се състави командна процедура на езика на командния интерпретатор bash за Linux, която получава при стартиране два параметъра: първият параметър е име на директория, а вторият параметър е символен низ от вида d--------- (9 символа), задаващ права на достъп.
Ако съществува директория със зададеното име и нейните права на достъп съвпадат с втория параметър, процедурата копира всички файлове от текущата директория, имащи размер, по-голям от 600 байта, в посочената директория.
В противен случай процедурата извежда на стандартния изход подходящо съобщение.

Задача 2 (3 т.)
Даден е следният фрагмент от програма на C, в който са използвани системни примитиви на ОС UNIX и LINUX:

```
int fd;
write ( 1, "START\n", sizeof("START\n" );
close ( 1 );
fd = creat ( "file.txt", 0666 );
write ( fd, "START\n", sizeof("START\n" );
write ( 1, "START\n", sizeof("START\n" );
```

Напишете вдясно какво и къде ще бъде изведено като резултат от изпълнението на този фрагмент.

Задача 3 (4 т.)
Даден е крайният недетерминиран автомат $A=<\{q_0,q_1,q_2,q_3,q_4,q_5,q_6\},\{a,b\},q_0,\delta,\{q_6\}>$ с функция на преходите δ:

$\delta(q_0,a)=\varnothing$	$\delta(q_2,a)=\{q_2,q_6\}$	$\delta(q_4,a)=\varnothing$	$\delta(q_6,a)=\varnothing$
$\delta(q_0,b)=\{q_1,q_3,q_6\}$	$\delta(q_2,b)=\{q_1\}$	$\delta(q_4,b)=\{q_4\}$	$\delta(q_6,b)=\varnothing$
$\delta(q_1,a)=\{q_2,q_6\}$	$\delta(q_3,a)=\{q_3\}$	$\delta(q_5,a)=\varnothing$	
$\delta(q_1,b)=\{q_5\}$	$\delta(q_3,b)=\{q_3,q_4,q_6\}$	$\delta(q_5,b)=\{q_1\}$	

Да се построи краен детерминиран автомат B, еквивалентен на автомата A.

Задача 4 (3 т.)
Какъв ще бъде резултатът от изпълнението на следния програмен фрагмент:

```
int M[2][3] = { 0 }, i , j , k = 0, N = 0 ;

for ( int * p = M, i = 0 ; ! * (p + i ) ; i = ( i + 5 ) % 6 )
      cout << * ( p + i ) = k++  ;
for ( i = 1 ; i ; i -- )
      for ( j = 1 ; j ; -- j )
            N = N * 10 + M[ i ][ j ] ;
cout << '\n' << N ;
```

Място за отговора

Задача 5 (2 т.)
Какъв ще бъде резултатът от изпълнението на следния програмен фрагмент:

```
void digit ( char v )
          { int d ;
            switch ( ( d = v – '0' ) * ( d > 0 ) * (d < 4 ) )
                    { default : cout << d << ' ';
                      case 0 : cout << "not" ;
                    }
          }
void main ( )
          { char word [5] = "3.14" ;
            for ( int p = 0 ; p <= 4 ; p++ )
                    { digit ( word [ p ] ) ; cout << '\n' ; }
          }
```

Място за отговора

Задача 6 (4 т.)
Какъв ще бъде резултатът от изпълнението на следния програмен фрагмент:

```
struct CN { char character ; CN *next ; } *first, cn ;

void list ( char * word )
          { cn.character = *word ; cn.next =  first ;
            first = new CN ; *first = cn ;
            if (*word++)  list (word) ;
          }
void print ( CN *first )
          { if (!first) {cout<<"\n result = " ; return ; }
            else  { print (first->next) ; cout<<first->character ; return ; }
          }
void main ( )
          { first = NULL ;
            list( "lover" ) ;
            print(first->next->next) ; }
```

Място за отговора

Задача 7 (3 т.)
Какъв ще бъде резултатът от изпълнението на обръщението R (5) ;
при следната дефиниция на функцията R:

```
void  R ( int r )
          { switch  ( ! ( r += ( r < 0 )  -  ( r > 0 ) )
                     { case 0  :  R ( - r ) ;
                       default :  cout <<  ' ' <<  r ;
          }
```

Място за отговора

Задача 8 (4 т.)

Изберете подходящи спецификатори за достъп и атрибут за област в йерархията base->der:

```
class base                          class der: …… base
{……: int b1;                       {……: int d1;
 ……: int b2;                        ……: int d2;
 ……: int b3();                      ……: int d3();
};                                  };
```

така, че фрагментите:

```
int der::d3()
{cout << b2 << "\n";
 cout << d1 << " " << d2 << "\n";
 return b2+d2;
};
der d1;
d1.d2 = 15;
d1.b2 = 25;
d1.d3();
```

да не предизвикват синтактични грешки, а фрагментът:

```
der d2;
d2.b1; d2.d1 = 22;
d2.b3();
```

да предизвиква синтактични грешки.

Задача 9 (8 т.)

Напишете резултата от изпълнението на програмата:

```
#include <iostream.h>
class A
{public:
 A(int a = 0)
 {n = a;
  x = 1.5;
  cout << "A.n::" << n << ", A.x::" << x << endl;
 }
 A(const A& p)
 {n = p.n;
  x = p.x;
  cout << "A.n::" << n << endl
              << "A.x::" << x << endl;
 }
 A& operator=(const A& p)
 {if(this!=&p)
    {n = p.n + 1;
     x = p.x + 1.5;
     cout << "A.n:: " << n << endl
                 << "A.x:: " << x << endl;
    }
  return *this;
 }
 private:
   int n;
   double x;
};
class B
{public:
 B(double b = 2.5)
 {n = 2;
  x = b;
  cout << "B.n::" << n << ",B.x::" << x << endl;
 }
 private:
   int n;
   double x;
};
```

```
class C
{public:
 C(double b = 3.5)
 {n = 3;
  x = b;
  cout << "C.n::" << n << ",C.x::" << x << endl;
 }
 C(const C& p)
 {n = p.n + 3;
  x = p.x + 3.5;
  cout << "C.n:: " << n << endl
            << "C.x:: " << x << endl;
 }
 private:
   int n;
   double x;
};
class D : protected B, A, public C
{public:
 D(int x = 1, int y = 2, int z = 3): A(x), B(y), C(z)
 {n = z;
  m = x*y;
  cout << "D.n::" << n << ",D.m::" << m << endl;
 }
 D& operator=(const D& p)
 {if(this!=&p)
    {A::operator =(p);
     n = p.n;
     m = p.m;
    }
  return *this;
 }
private:
  int n, m;
};
void main()
{D d1, d2(1,2,3), d3(4, 5);
 D d4 = d1;
 d3 = d2;
}
```

Задача 10 (общо 15 т.)

Клас DLList реализира свързан списък с елементи от тип int, представен с две връзки:

```
struct elem_link2
{int inf;                       // информационно поле
 elem_link2 *pred,              // указател към предишен елемент
            *succ;              // указател към следващ елемент
};
class DLList
{public:
 // член-функции на голямата четворка
 DLList();
 ~DLList();
 DLList(DLList const&);
 DLList& operator=(DLList const &);
 void print();                  // извеждане на списъка от началото към края
 void print_reverse();          //извеждане на списъка от края към началото
  // Установяване на итератора CurrentS в указана от p позиция или
  // по подразбиране в началото на списъка
 void IterStart(elem_link2*p = NULL);
  // Установяване на итератора CurrentE в указана от p позиция или
  // по подразбиране в края на списъка
```

```
    void IterEnd(elem_link2*p = NULL);
    // Преместване на итератора CurrentS в следващата позиция
       elem_link2* IterSucc();
    // Преместване на итератора CurrentE в предходната позиция
      elem_link2* IterPred();
    // Включване на елемента x в края на списък
      void ToEnd(int const & x);
    // Изключване на указан от указателя p елемент от подразбиращия
    // се св.списък. Елементът се записва в параметъра x.
    void DeleteElem(elem_link2* p, int & x);
     private:
     elem_link2 *Start,                    // указател към началото на списъка
                        *End,              // указател към края на списъка
                        *CurrentS,         // итератор за начало
                   *CurrentE;              // итератор за край
     void DeleteList();                    // изтрива подразбиращия се списък
     void CopyList(DLList const &r);       // копира списъка r в
                                           // подразбиращия се списък
   };
```

а) **(2 т.)** Да се дефинират член-функциите на голямата четворка:

б) **(4 т.)** Да се дефинират член-функциите
void CopyList(DLList const & r); и void DeleteList();

в) **(2 т.)** Да се дефинира член-функцията
void DeleteElem(elem_link2* p, int & x);

г) **(3 т.)** Да се дефинира външна булева функция, която проверява дали свързан списък от цели числа, представен чрез две връзки, е симетричен относно средата си:

д) **(4 т.)** Свързан списък, съдържащ 2n цели числа, е представен чрез две връзки така, че първите му n елемента са сортирани във възходящ, а вторите n елемента – в низходящ ред. Да се напише външна процедура, която чрез сливане сортира във възходящ ред елементите на списъка:

Задача 11 (общо 15 т.)

Шаблонът на класа tree реализира двоично дърво с елементи от тип T:

```
template <class T>
struct node                 // тройна кутия
{T inf;                     // информационно поле
 node *Left,                // указател към ляво поддърво
      *Right;               // указател към дясно поддърво
};
template <class T>
class tree
{public:
 tree();                            // конструира празно двоично дърво
 ~tree();                           // деструктор
 tree(tree const&);                 // конструктор за присвояване
 tree& operator=(tree const&); // оператор =
 bool empty() const;                // проверява дали двоично дърво е празно
 T RootTree() const;                // намира корена на двоично дърво
 tree LeftTree() const;             // намира лявото поддърво на двоично дърво
 tree RightTree() const;            // намира дясното поддърво на двоично дърво
// Create 3  създава непразно дв. дърво по зададени корен,
// ляво и дясно поддърво
 void Create3(T const&, tree<T> const&, tree<T> const&);
 void print() const;                    // извежда двоично дърво
 void Create();                         // създава непразно двоично дърво
 // …
 private:
  node<T> *root;                        // указател към тройна кутия
  // DeleteTree - изтрива дървото, указано от указателя p
  void DeleteTree(node<T>* &p) const;
```

```
 // Copy - копира дървото, указано от указателя q на друго място в
 // паметта, указано от указателя p
  void Copy(node<T> * &p, node<T>* const&q) const;
 // CopyTree - копира дървото t в подразбиращото се дърво
  void CopyTree(tree<T> const & t);
 // …
};
```

а) **(3 т.)** Реализирайте член-функциите: tree& operator=(tree const&); T RootTree() const;
и tree LeftTree()const; на шаблона на класа tree:

б) **(3 т.)** Реализирайте член-функцията
void Copy(node<T>*& p, node<T>* const& q) const; на шаблона на класа tree:

в) **(3 т.)** Реализирайте член-функцията
void DeleteTree(node<T>* &p) const; на шаблона на класа tree:

г) **(3 т.)** Едно двоично дърво е по-малко (<) от друго, ако е негово поддърво. Предефинирайте оператора < чрез шаблон на функция-приятел на шаблона на класа tree:

д) **(3 т.)** Дефинирайте шаблон на външна функция, която прилага към всеки от върховете на дадено двоично дърво с елементи от тип T функцията f: T → T

Задача 12 (4 т.)
Коя от изброените програми на Prolog пресмята функцията факториел (т.е. за всяко естествено число n>0 при цел **? – fact(n, Y).** връща в Y стойността на n!)?

а) fact(X, Y) :- fact(X–1, Z), Y is Z*X.
   fact(1,1).
б) fact(X, Y) :- X1 is X–1, fact(X1, Z), Y is Z*X.
   fact(1,1).
в) fact(1, 1).
   fact(X, Y) :- X > 0, X1 is X–1, Y is Z*X, fact(X1, Z).
г) fact(X, Y) :- fact_1(X, 1, Y).
   fact_1(1, Y, Y).
   fact_1(X, Y, Z) :- X1 is X–1, Y1 is X*Y, fact_1(X1, Y1, Z).

Задача 13 (3 т.)
Намерете най-общ унификатор (НОУ) на изразите p(X, Y, Z) и p(f(Y), g(Z), h(X)) или докажете, че такъв НОУ не съществува.

Задача 14 (5 т.)
Нека P(f) и Q(f) са непрекъснати свойства в множеството на всички едноместни частични функции. За кои от изброените свойства може да се твърди, че също са непрекъснати:
а) $P \vee Q$ б) $P \wedge Q$
в) $P \Rightarrow Q$ г) $\neg P$

Задача 15 (4 т.)
Нека R е следната рекурсивна програма над типа данни Nat:
R: F(X, X) where
   F(X, Y) = if X = 0 then Y else F(X–1, G(X, Y))
   G(X, Y) = if Y = 0 then X else G(X, Y–1).

Кои от изброените условия са верни за D_V(R) (денотационната семантика на R с предаване на параметрите по стойност):
а) $\forall$ x (!D_V(R)(x) $\Rightarrow$ D_V(R)(x) = 0)
б) $\forall$ x (!D_V(R)(x) $\Rightarrow$ D_V(R)(x) = x!)
в) $\forall$ x (x > 0 $\Rightarrow \neg$!D_V(R)(x))
г) $\forall$ x ((x > 0 & !D_V(R)(x)) $\Rightarrow$ D_V(R)(x) = 1)

Задача 16 (2 т.)
Дефинирайте примерна функция на езика Scheme, която генерира логаритмичен рекурсивен процес:

Обяснете каква задача се решава с така дефинираната функция: .....
.....

Задача 17 (общо 8 т.)
Дадено е дърво, възлите на което за означени с уникални символни атоми. Дървото е представено чрез асоциативен списък, ключове в който са възлите на дървото, а асоциирана с даден ключ стойност е списъкът от преките наследници на съответния възел от дървото.
а) **(3 т.)** Дефинирайте функция на езика Scheme, която връща като резултат "дядото" в дървото (родителя на родителя) на даден възел ***node***:

б) **(5 т.)** Дефинирайте функция на езика Scheme, която връща като резултат списък от всички възли на дървото, които са наследници (преки или непреки) на даден възел ***node***:

Задача 18 (общо 7 т.)
Базата от правила и работната памет (базата от факти) на една система, основана на правила, са описани със средствата на езика Prolog както следва:

```
% Дефиниции на използваните оператори
:- op(900,fx,if).
:- op(890,xfy,then).
:- op(880,xfy,or).
:- op(870,xfy,and).

% Правила
if has_feathers(X) then bird(X).
if has_gills(X) then fish(X).
if has_hair(X) or gives_milk(X) then mammal(X).
if bird(X) and has_long_neck(X) then swan(X).
if mammal(X) and has_long_neck(X) then giraffe(X).
if mammal(X) and can_speak(X) then human(X).
if bird(X) and can_speak(X) then parrot(X).
if can_speak(X) then intelligent(X).
```

```
% Факти (работна памет)
fact(has_hair(gogo)).
fact(has_feathers(bobo)).
fact(has_feathers(pepo)).
fact(has_gills(fifi)).
fact(can_speak(bobo)).
fact(can_speak(gogo)).
fact(has_long_neck(pepo)).
```

а) **(3 т.)** Предполага се, че интерпретаторът на правилата извършва прав извод (извод, управляван от данните), като на всяка стъпка от работния си цикъл записва заключението на избраното за изпълнение правило като нов факт в края на работната памет. В конфликтното множество се включват само правила, при чието изпълнение се записват нови факти в работната памет.

Какво ще бъде съдържанието на работната памет след завършване на работата на интерпретатора на правилата, ако на всяка стъпка от работния си цикъл той избира и изпълнява първото правило от конфликтното множество?

б) **(4 т.)** Предполага се, че интерпретаторът на правилата извършва обратен извод (извод, управляван от целите), осъществяван от отношението is_true(Goal,List), като на всяка стъпка от работния си цикъл записва в списъка List кое правило (или факт / отговор на зададен от интерпретатора уточняващ въпрос) е било използвано, за да се достигне до необходимото заключение (текущата цел).

Допишете дефиницията на отношението is_true:

```
is_true(P,[fact(P)]):- fact(P), !.
is_true(P,[rule(if Condition then P)|L]):- ……., is_true( ..……... , ……. ).
is_true(P and Q,L):- is_true(P,L1), is_true(Q,L2), append( ….…… ,  …….. , …… ).
is_true(P or Q,L):- …….. ; …… .
is_true(P,[asked(P)]):-
    \+ (if _ then P),
    \+ (P = (_ and _)),
    \+ (P = (_ or _)),
    write('Is '), write(P), write(' true (y/n): '),
    …… ,
    ……... .
```

Задача 19 (общо 5 т.)

Създадена е база от данни ***BookStores*** за малка верига от книжарници. В базата от данни се съхранява информация за книги (***books***), автори (***authors***) и издателствата, с които работят (***publishers***). Съхраняват се данни и за отделните магазини (***stores***) и служителите в тях (***employees***). В базата от данни се пази отчет за извършените продажби на книги (***sales_profit***). Релационната схема на базата от данни е описана с релационните схеми (фиг. 1):

- **Stores** – съхранява информация за отделните магазини: град (***city***), адрес (***address***), пощенски код (***zip***), телефонен номер (***phone_number***). Всеки един магазин от веригата се идентифицира с уникален номер (***store_id***).

stores(store_id, city, address, zip, phone_number)

- **Employees** – съхранява информация за отделните служители в книжарниците: имена (***first_name***, ***last_name***), длъжност (***job_type* – seller** / **manager**), идентификатор на магазин, в който работят (***store_id***), заплата (***salary***), дата на назначаване на длъжност (***hire_date***), както и лични данни като телефонен номер (***phone_number***) и идентификатор на мениджър (***manager_id***). Всеки един служител се идентифицира с уникален номер (***emp_id***). Всеки продавач има само един мениджър. Мениджърите от най-високото ниво в йерархията нямат мениджър.

employees(emp_id, job_type, store_id, first_name, last_name, phone_number, salary, hire_date, manager_id)

- **Sales_profit** – съхранява информация за продажби на книги: служител (***emp_id***) е продал книга (***book_id***) на дата (***date***).

sales_profit(book_id, emp_id, date)

- **Publishers** – съхранява информация за отделните издателства, с които книжарниците работят: име (***pub_name***), държава (***country***), град (***city***), адрес (***address***), пощенски код (***zip***) и телефонен номер за връзка (***phone_number***). Всяко издателство се идентифицира с уникален номер (***pub_id***).

publishers(pub_id, pub_name, country, city, address, zip, phone_number)

- **Authors** – съхранява кратка информация за отелните автори на книги: имена (***first_name***, ***last_name***) и националност (***nationality***), както и уникален номер за всеки автор (***author_id***).

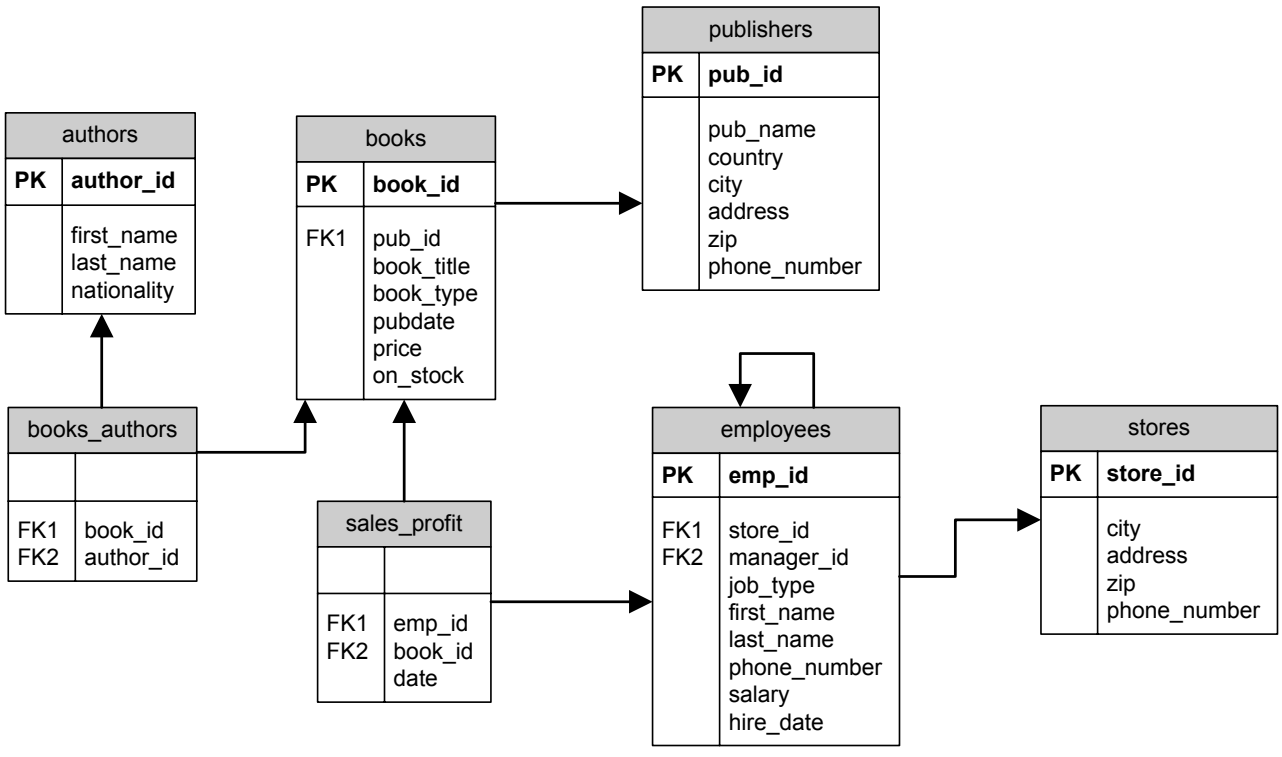
authors(author_id, first_name, last_name, nationality)

- **Books** – съхранява информация за отделните книги: заглавие (***book_title***), тип (***book_type***), идентификатор на издателството (***pub_id***), дата на издаване (***pubdate***), цена (***price***) и налични бройки на склад (***on_stock***). Всяка книга в базата се идентифицира с уникален номер (***book_id***).

books(book_id, book_title, book_type, pub_id, pubdate, price, on_stock)

- **Books_authors** – съхранява информация за авторите на отделните книги: идентификатор на книга (***book_id***), идентификатор на автор (***author_id***).

books_authors(book_id, author_id)



Фиг. 1. Релационна схема на база от данни ***BookStores***

В рамките на една select заявка:

а) **(3 т.)** За всеки мениджър с брой на продажби на подчинените му служители, по-голям от 50, да се изведе информация за името му, град, в който работи, и брой на продажбите на подчинените му служители.

б) **(2 т.)** Да се изведат заглавията на издателствата, градовете, в които имат продажби, и сумата от продажбите им.