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

ДЪРЖАВЕН ИЗПИТ
ЗА ПОЛУЧАВАНЕ НА ОКС "БАКАЛАВЪР ПО ИНФОРМАТИКА"
11-12.07.2006 г.

ЧАСТ I (ПРАКТИЧЕСКИ ЗАДАЧИ)

Задача 1 (5 т.)

Да се състави командна процедура на езика на командния интерпретатор bash за Linux, която получава при стартиране два параметъра: първият – име на файл, съдържащ списък от идентификатори на потребители на системата, всеки от които на отделен ред, вторият – символен низ.

Ако не съществува директория с име, съвпадащо с втория параметър, процедурата я създава, записва в нея новосъздаден файл, достъпен за четене и писане само за собственика и съдържащ само онези от идентификаторите в зададения списък, които не са на потребители, работещи в сесия, а останалите имена от списъка извежда на стандартния изход.

Задача 2 (4 т.)

Даден е следният фрагмент от програма на езика С, в който са използвани системни примитиви на ОС UNIX и LINUX:

```
int  pid, flag=0, status;

if ( (pid  = fork() ) = = 0 ) flag++;
else {wait( &status); exit(0);}
++flag;
printf( " Stoinostta na flag = %d;", flag );
```

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

а) Stoinostta na flag =1;
б) Stoinostta na flag =2; Stoinostta na flag =1;
в) Stoinostta na flag =2;
г) Stoinostta na flag=1;Stoinostta na flag =2;

Моля, заградете верния отговор с кръгче.

Задача 3 (2 т.)

Текстов файл с име **proc** съдържа следната последователност от команди на bash за Linux:

```
if [ $#  -gt  3  ]
then  echo $#
else count=0
fi
for val
do
  if  [  "$val"   !=   "2"  ]
  then
     count=`expr $count + 1 `
  fi
done
echo $count
```

Какъв резултат ще бъде изведен на стандартния изход при стартиране на изпълнение чрез командния ред:

```
sh  proc  1  2  3
```

а) 3
1

б) 3

в) 3
2

г) 1

д) 2

е) 3
3

Моля, заградете верния отговор с кръгче.

Задача 4 (общо 10 т.)

Дадено е множеството от двоични функции A={x⊕y, xy⊕z, x⊕y⊕z⊕1, xy∨yz∨zx}.

а) (2 т.) Да се представи в табличен вид всяка от функциите от множеството A:

x	y	z	x⊕y	xy⊕z	x⊕y⊕z⊕1	xy∨yz∨zx

б) (5 т.) За всяка от функциите от множеството A да се изследва принадлежността ѝ към всяко от множествата T_0, T_1, S, M и L и да се попълни в съответната клетка от таблицата знак "+", ако функцията принадлежи на съответното множество, и знак "–" в противния случай:

Функция	T_0	T_1	S	M	L
$x \oplus y$					
$xy \oplus z$					
$x \oplus y \oplus z \oplus 1$					
$xy \vee yz \vee zx$					

в) (3 т.) Ако множеството A е пълно, да се намерят всички базиси:

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

Място за отговора

Задача 6 (2 т.)
Какъв ще бъде резултатът от изпълнението на следния програмен фрагмент:

```
void letter ( char v )
          { int d ;
            switch ( ( d = v – 'a' + 1) * ( d > 0 ) * (d < 4 ) )
                    { default : cout << "Not " ;
                      case 1 : cout << 'A' ;
                      case 2 : cout << 'B' ;
                      case 3 : cout << 'C' ;
                    }
          }
void main ( )
          { char word [5] = "Abac" ;
            for ( int p = 0 ; p <= 4 ; p++ )
                    { letter ( word [ p ] ) ; cout << '\n' ; }
          }
```

Място за отговора

Задача 7 (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+1) ;
          }
void print ( CN *first )
          { if(!first) {cout<<"\n result = " ; return ; }
            else  {print(first->next) ; cout<<first->character ; return ; }
          }
void main ( )
          { first = NULL ;
            list( "droll" ) ;
            print(first->next->next) ; }
```

Място за отговора

Задача 8 (3 т.)
Какъв ще бъде резултатът от изпълнението на обръщението R (5) ;
при следната дефиниция на функцията R:

```
void  R ( int r )
          { if  ( r -=  r > 0  -  r < 0 )  R ( - r ) ;
             cout <<  ' ' <<  r ;
          }
```

Място за отговора

Задача 9 (общо 12 т.)

Шаблонът на класа LList реализира свързан списък, представен чрез една връзка.

```
template <class T>
struct elem_link1
{T inf;
 elem_link1<T> *link;
};
template <class T>
class LList
{public:
  LList();
  ~LList();
  LList(LList const&);
  LList& operator=(LList const &);
  void IterStart(elem_link1<T>* = NULL);
  elem_link1<T>* Iter();
  void InsertAfter(elem_link1<T>* p, const T& x);
  // включва x  след указания от p елемент
  bool DeleteBefore(elem_link1<T> *p, T & x);
  // изтрива елемента пред сочения от указателя p и го записва в x
  //…….
 private:
  elem_link1<T> *Start,               // указател към началото
                *End,                 // указател към края
                 *Current;            // итератор
// ……
  };
```

а) (1 т.) Член-функцията

```
void IterStart(elem_link1<T>* = NULL);
```

установява итератора в указания чрез параметъра адрес, ако той е ненулев, и в началото на списъка - в противен случай. Реализирайте я!

б) (1 т.) Член-функцията

```
elem_link1<T>* Iter();
```

премества итератора в следваща позиция (ако е възможно), като връща предишната му позиция. Реализирайте я!

в) (2 т.) Член-функцията

```
void InsertAfter(elem_link1<T>* p, const T& x);
```

включва елемента x след указания от указателя p елемент на подразбиращия се списък. Реализирайте я!

г) (2 т.) Член-функцията

```
bool DeleteBefore(elem_link1<T> *p, T & x);
```

изтрива, ако е възможно, елемента пред сочения от указателя p и го записва в x. Реализирайте я!

д) (2 т.) Нека
```
typedef LList<int> IntList;
```
Като използвате член-функциите IterStart, Iter, DeleteElem на класа IntList, реализирайте външната функция:
```
void print_reverse(IntList L);
```
която извежда на екрана в обратен ред елементите на списъка от цели числа L.

е) (2 т.) Нека L е свързан списък с елементи от тип T, а f е едноаргументна функция от вида: f: T → T. Дефинирайте шаблон на функция от по-висок ред map:
```
LList<T> map(T (*f)(T), LList<T>&L);
```
която прилага f над всеки от елементите на L и връща получения списък.

ж) (2 т.) Предефинирайте оператора == чрез шаблон на функция-приятел на шаблона на класа LList<T>.

Задача 10 (общо 8 т.) Шаблонът на класа BinOrdTree реализира двоично-наредено дърво 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 operator==(BinOrdTree<T> const& t);
    bool empty()const;              // проверява дали дървото е празно
    T RootTree() const;             // връща корена на двоично-наредено дърво
    BinOrdTree LeftTree() const;    // връща лявото поддърво
    BinOrdTree RightTree() const;   // връща дясното поддърво
    void AddNode(T const & x)       // включва указания елемент
    { Add(root, x);
    }
    void DeleteNode(T const&);      // изтрива указания елемент
    void Create3(T const&,BinOrdTree<T> const&, BinOrdTree<T> const&);
 // ……….
```

```
 private:
    node_bin<T> *root;
  // DeleteTree изтрива двоично-наредената дърво, зададено чрез указателя p
   void DeleteTree(node_bin<T>* &p) const;
    void CopyTree(BinOrdTree<T> const&);
 // CopyTree копира указаното в неявното двоично-наредено дърво
    void Copy(node_bin<T> * &, node_bin<T>* const&) const;
void Add(node_bin<T> *&, T const &) const;
 // ……
};
```

а) (2 т.) Дефинирайте следните член-функции на шаблона:

```
~BinOrdTree();
void DeleteTree(node_bin<T>* &) const;
```

б) (1,5 т.) Член-функцията void AddNode(T const & x) на шаблона на класа включва указания като параметър елемент в неявното двоично-наредено дърво. Тя използва помощната член-функция
void BinOrdTree<T>::Add(node_bin<T>* &p, T const & x) const, която включва елемента x в двоично-наредената дърво, зададено чрез указателя p.
Реализирайте член-функцията Add!

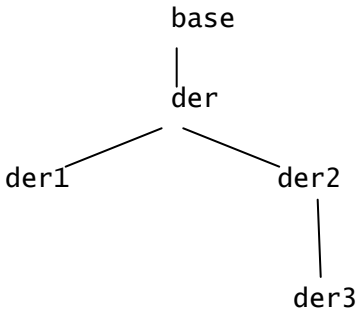
в) (1,5 т.) Реализирайте член-функцията

```
bool operator==(BinOrdTree<T> const& t);
```

на шаблона BinOrdTree<T>.

г) (3 т.) Проверете дали дадено двоично-наредено дърво с елементи от тип T е идеално балансирано (всеки негов връх има ляво и дясно поддърво, броят на възлите на които се различава най-много с 1).

Задача. 11 (6 т.) Посочете и коментирайте грешките в дефинициите на конструкторите в следната йерархия:




```
#include <iostream.h>
class base
{private: int a1;
 public:
 void read(int x = 0)
 {a1 = x;
 }
 void display()
 {cout << "a1: " << a1 << endl;
 }
};
class der : public base
{private: int d1;
 public:
  der(int x) : der(x=0);
 void display()
 {cout << "d1: " << d1 << endl;
  cout << "base::display():" << endl;
  base::display();
 }
};
```

```
der::der(int x): base(x=5)
{cout << "constructor der\n";
 d1 = x;
}
class der1 : public der
{private: int d1;
 public:
 void display()
 {cout << "der1 - member d1: " << d1 << endl;
  cout << "der::display():" << endl;
  der::display();
 }
};
class der2 : public der
{private: int d1;
 public:
  der2()
  {d1=0;
  }
  der2(int x) : der(x=0);
  void display()
  { cout << "der2 - member d1: " << d1 << endl;
    cout << "der::display():" << endl;
    der::display();
  }
};
der2::der2(int x) : base(x)
{cout << "constructor der2\n";
 d1 = x;
}
class der3 : public der2
{private: int d1;
 public:
 void display()
 {cout << "der3 - member d1: " << d1 << endl;
  cout << "der1::display():" << endl;
  der1::display();
 }
};
void main()
{der a(1); a.display();
 der1 b; b.display();
 der2 c(1); c.display();
 der3 d; d.display();
}
```

Задача 12 (8 т.) Задраскайте грешните обръщения към функции в главната функция *main* на програмата. Какъв е резултатът от изпълнението ѝ след отстраняване на грешките? (*Напишете резултата отстрани на програмата*).

```
#include <iostream.h>
class Base
{public:
   void help()
    {cout << "help()\n";
     f();
     h();
     g();
    }
   virtual void f()
    {cout << "f()\n";
    }
 protected:
   virtual void h()
    {cout << "h()\n";
   }
 private:
   virtual void g()
   {cout << "g()\n";
   }
};
class Der1 : public Base
{protected:
   virtual void f()
   {cout << "Der1 class\n";
    Base::f();
   }
 public:
   virtual void h()
   {cout << "Der1-h()\n";
   }
   virtual void g()
   {cout << "Der1-g()\n";
   }
};
class Der2 : public Base
{  virtual void h()
   {cout << "Der2 class\n";
    Base::h();
   }
 protected:
   virtual void g()
   {cout << "Der2-g()\n";
   }
```

```
   virtual void f()
   {cout << "Der2-f()\n";
   }
};
void main()
{ Base *x = new Base;
  Base *y = new Der1;
  Der2 o;
  Base *z = &o;
  x->f();               y->f();            z->f();
  x->h();               y->h();            z->h();
  x->g();               y->g();            z->g();
  Der1 *p = new Der1;
  Der2 *q = new Der2;
  o.f();                 p->f();           q->h();
  o.g();                 p->g();           q->h();
  x->help();             y->help();   z->help();
  o->help();             p->help();
  delete x; delete y; delete p; delete q;
}
```

Задача 13 (3 т.)
Кои от изброените предикати са генератори за множеството на естествените числа
N = {0,1,2,…} (т.е. при цел ? – nat(X). при преудовлетворяване пораждат последователно елементите на N):
а) nat(0).
nat(X) :- nat(X-1).
б) nat(0).
nat(X) :- X is Y+1, nat(Y).
в) nat(0).
nat(X) :- nat(Y), X is Y+1.
г) nat(0).
nat(X) :- Y is X – 1, nat(Y).

Задача 14 (4 т.)
Нека F е съвкупността от всички едноместни частични функции в множеството N на естествените числа. Кои от изброените оператори Γ_i: F $\longrightarrow$ F, $1 \le i \le 4$ са монотонни? Обосновете отговорите си:

а) $\Gamma\_1(f)(x) \cong$ if $x \equiv 0 \pmod 2$ then $x + 1$ else $x - 1$
б) $\Gamma\_2(f)(x) \cong$ if $x \equiv 0 \pmod 2$ then $x + 1$ else $x + 2$
в) $\Gamma\_3(f)(x) \cong$ if f е тотална then недефинирано else 1
г) $\Gamma\_4(f)(x) \cong$ if f е тотална then 1 else недефинирано.

Задача 15 (3 т.)
Кои от операторите от предишната задача имат най-малка неподвижна точка f_Γ
(ако f_Γ съществува, напишете коя е; ако не – посочете защо не съществува):

Задача 16 (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(Y))
G(X) = if X = 0 then 0 else G(X – 1) + 2.

Кои от изброените условия НЕ са верни за D_V(R) (денотационната семантика на R с предаване на параметрите по стойност):
а) $\forall$ x ((x > 0 & ! D_V(R)(x)) $\Rightarrow$ D_V(R)(x) $\equiv$ 0 (mod x));
б) $\forall$ x (! D_V(R)(x) $\Rightarrow$ D_V(R)(x) $\equiv$ 0 (mod 2));
в) $\forall$ x ((x > 0 & !D_V(R)(x)) $\Rightarrow$ D_V(R)(x) > 2^x);
г) $\forall$ x (! D_V(R)(x)).

Задача 17 (5 т.)
Дефинирайте функция на езка Scheme, която връща като резултат сумата от естествените числа в интервала ***[a,b]*** (***a*** и ***b*** са две дадени естествени числа, ***a≤b***), които са кратни на 3 и в десетичния запис на които се съдържа цифрата 3, като за целта реализира итеративен изчислителен процес:

Задача 18 (4 т.)
Напишете оценката на всеки от следващите изрази на езика Scheme:
(cdr (cadr '((a (b)) ((c (d)) e)))) → ……
(cons '(a b) (list 'c '((d) e))) → ……
((lambda (x) (cons x (list 1 2))) 'a) → ……
(append (map list '(a b c)) (map list '(x y))) → ……

Задача 19 (5 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,e). arc(a,g).
arc(b,d). arc(b,g). arc(c,d). arc(d,e). arc(e,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(s,5). h(a,4). h(b,3). h(c,2).
h(d,4). h(e,1). h(g,0).

Ако се търси път в графа от върха "s" до върха "g", попълнете следната таблица:

Стратегия за търсене	Намерен път	Обходени възли
Depth-first search		
Best-first search		
Hill Climbing		

Задача 20 (2 т.)
Дадени са правилата R1 и R2 и фактите P1, P2, P3, P4 и P5. Какви ще бъдат факторите за сигурност (Certainty Factors – CF) на C1 и C2 след изпълнението на правилата?
R1: IF ((P1 and P2) or P3) or (P4 and P5) THEN C1 : 0.6
R2: IF (P2 and P4) or (C1 and P3) THEN C2 : 0.5
CF(P1)=0.6 CF(P2)=0.5 CF(P3)=0.3 CF(P4)=0.8 CF(P5)=0.5

Задача 21 (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***).

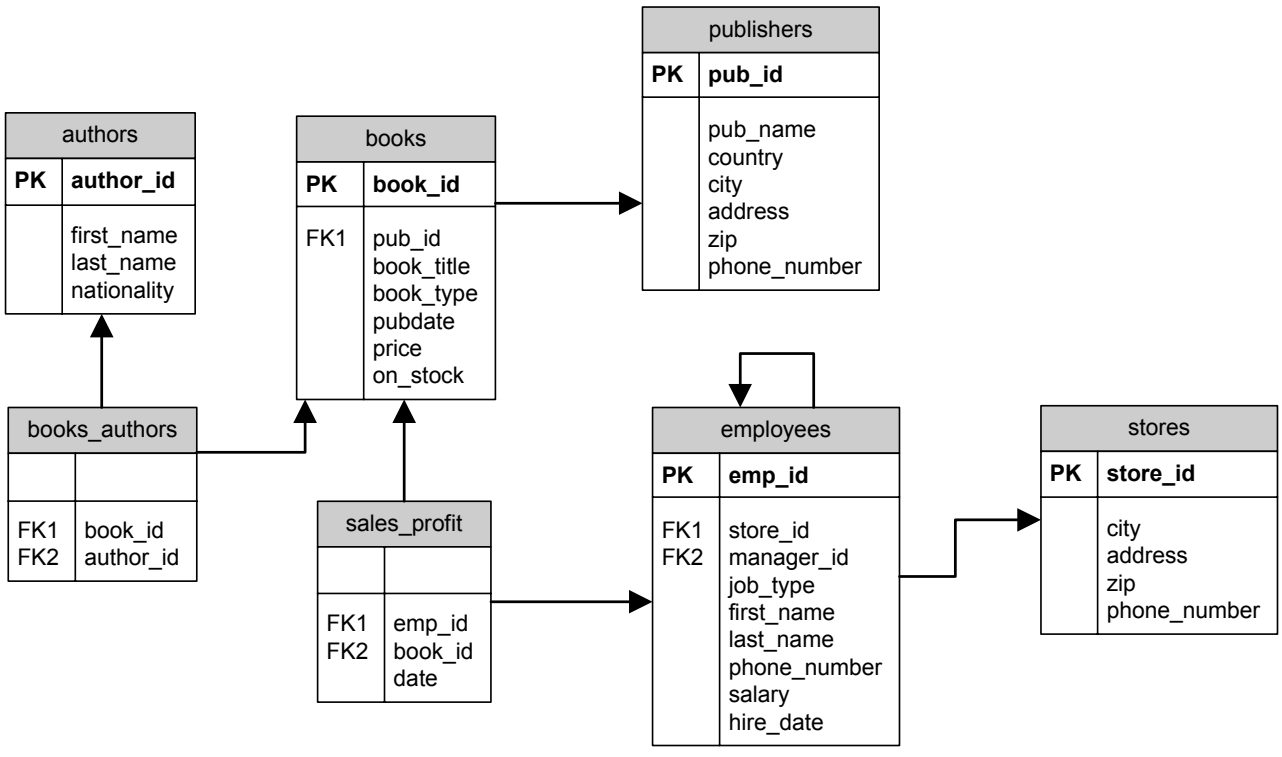
authors(author_id, first_name, last_name, nationality)

- **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 обща сума на заплатите на подчинените му служители, по-голяма от 1200 лева, да се изведе информация за името му, град, в който работи, брой на подчинените му служители и сума на заплатите на подчинените му служители.

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