СОФИЙСКИ УНИВЕРСИТЕТ
"СВ. КЛИМЕНТ ОХРИДСКИ"

ФАКУЛТЕТ ПО МАТЕМАТИКА
И ИНФОРМАТИКА

ДЪРЖАВЕН ИЗПИТ

ЗА ПОЛУЧАВАНЕ НА ОКС "БАКАЛАВЪР ПО ИНФОРМАТИКА"

15 - 16.09.2007 г.

ЧАСТ I (ПРАКТИЧЕСКИ ЗАДАЧИ)

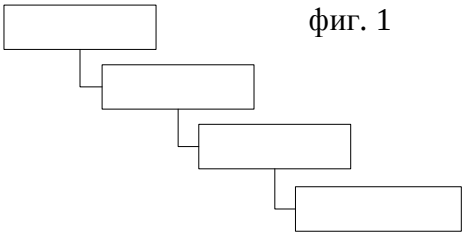
Задача 1. (6 т.) Напишете вдясно какво ще се извежда на стандартния изход (терминала) при изпълнението на следната последователност от команди на интерпретатора **bash** за LINUX

```
br=0
for var in a1 a2 a3
do
      set $var
done
echo $1 $2 $3
listpar=`echo $*`
echo $listpar
for each
do
      echo $each > filepar
      br=`expr $br + 1`
done
cat filepar
shift
echo $# $br
if [ $# -eq `expr $br - 1` ]; then
      set a2
      until  cat filepar | grep "$1"
      do
            echo br=$br
            break
      done
      echo END1
else
      echo END2
fi
```

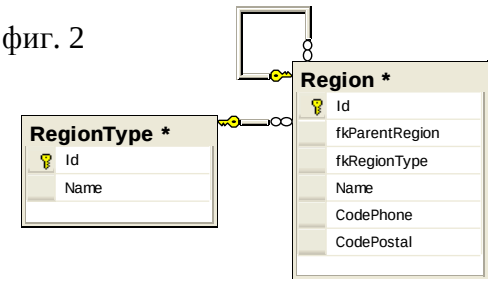
Решение:

Задача 2. (5 т.) За съхраняване на регионалната структура на България е създадена база от данни **Regions**. Структурата на регион е йерархична, както е показано на фиг. 1

фиг. 1



фиг. 2



Релационната схема на базата от данни е (фиг. 2):

Region(**Id**, fkParentRegion, fkRegionType, Name, CodePhone, CodePostal)
RegionType (**Id**, Name)

където:

* Таблица / *Колона*	*Описание*	*Тип на данните*	*Ограничения*
*** Region**	Регион от административното деление		
Id	Генерира се автоматично	int	PK
fkParentRegion	Йерархичен родител - **Region.Id**	int	FK
fkRegionType	Тип на региона - **RegionType.Id**	int	FK
Name	Име	varchar(50)	
CodePhone	Телефонен код	varchar(10)	
CodePostal	Пощенски код	varchar(10)	
*** RegionType**	Тип на региона		
Id	Генерира се автоматично Уникално идентифицира типа на региона	int	PK
Name	Име на региона: - държава - област - окръг - населено място (град) - населено място (село) - населено място (махала)	varchar(20)	

Легенда	
PK	Първичен ключ (Primary Key)
FK	Вторичен ключ (Foreign Key)

Да се изведе информация за областта с най-малко населени места. Справката да има вида

Име на област	Брой населени места в областта

Решение:

Задача 3. (6 т.) Да се определи за кои стойности на параметъра **a** и на броя на променливите $\mathbf{n \geq 2}$ е шеферова следната двоична функция:

$$\mathbf{f(\tilde{x}^n) = a \oplus (x_1 \mid x_2) \oplus (x_2 \mid x_3) \oplus \ldots \oplus (x_{n-1} \mid x_n) \oplus (x_n \mid x_1)}$$

Задача 4. (7 т.) Езикът **L** е описан с регулярния израз **xy(x+y)*x(x+y)*+yx(x+y)*y(x+y)*.**

a) Да се построи детерминиран краен автомат **A**, който разпознава езика **L** ;

b) Да се построи минимален детерминиран краен автомат **B**, еквивалентен на автомата **A**.

ЧЕРНОВА ЗА ЗАДАЧИ 3 и 4

Задача 5. (3 т.) Кой от изброените предикати **nat** е генератор на множеството на естествените числа **N = { 0, 1, 2, … }**, т.е. при цел **?- nat(X).** поражда последователно в **X** (при преудовлетворяване) елементите на **N**.

a)
```
nat(0).
nat(X):- nat(X–1).
```
b)
```
nat(0).
nat(X):- X is Y+1, nat(Y).
```
c)
```
nat(0).
nat(X):- nat(Y), X is Y+1.
```
d)
```
nat(0).
nat(X):- Y is X–1, nat(Y).
```

Задача 6. (3 т.) Кой от изброените оператори е монотонен, но не е компактен?

a) `Γ( f)( x) ≅ if x = 0 then 1 else 2.f( x – 1);`
b) `Γ( f)( x) ≅ if ¬!f( x) then x else f( x);`
c) `Γ( f)( x) ≅ f( x + 1);`
d) `Γ( f)( x) ≅ if (f е крайна) then недефинирано else 0.`

Задача 7. (6 т.) Нека **Γ** е следният оператор, преработващ двуместни частични функции в множеството на естествените числа:

```
Γ( f )( x, y) ≅ if  x = 0 then  y
                          else  if  y = 0  then  f( x – 1, 1)
                                           else  f( x - 1,  f( x, y – 1)).
```

Докажете, че за най-малката неподвижна точка f_Γ на оператора **Γ** е изпълнено условието:

$$\forall x\ \forall y\ (x > 0\ \&\ !f_\Gamma(x, y) \rightarrow f_\Gamma(x, y) \cong 1).$$

ЧЕРНОВА ЗА ЗАДАЧИ 5, 6 и 7

Задача 8. (3 т.) Дадени са следните изрази:

```
(1)    ((lambda (x y) (x (y 4))) (lambda (y) (* y 5)) sqrt)
(2)    (cadr (list '(() (16 (0))) (caadr '(((14 10) 2) (1 (11)) ((8) (9))))))
(3)    (let* ((x (lambda (x) (* x x x)))
              (x (x 5)))
              (* 2 (quotient x 25)))
```

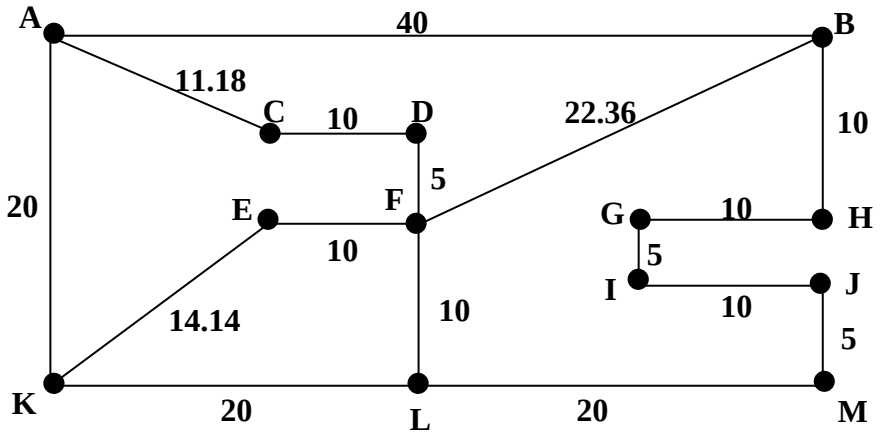
Посочете, кое от следните твърдения е вярно:

- **a)** Изразите (1) и (2) имат еднаква оценка
- **b)** Изразите (1) и (3) имат еднаква оценка
- **c)** Изразите (2) и (3) имат еднаква оценка
- **d)** Изразите (1), (2) и (3) имат еднаква оценка
- **e)** Нито едно от предходните твърдения не е вярно

Задача 9. (4 т.) Да се напише процедура на Scheme, която генерира линейно итеративен процес за пресмятане на корен квадратен от дадено положително число **x** с дадена точност **eps** по метода на Нютон, ако е дадена рекурентната зависимост за получаване на следващото приближение:

$$y_n = \begin{cases} x & ,n = 0 \\ \frac{1}{2}\left(y_{n-1} + \frac{x}{y_{n-1}} \right) & ,n > 0 \end{cases}$$

Задача 10. Даден е неориентиран граф (фиг. 2), като за всяка дъга е указано нейното тегло и за всеки възел е зададена стойността на евристичната функция **h (Таблица 1),** определяща разстоянието от дадения възел до целевия възел **M (Goal)**. Като се използва алгоритъма **A*** да се намери път от върха **A (Start)** до върха **M (Goal)**.



фиг. 2

	A	B	C	D	E	F	G	H	I	J	K	L	M
h	90	10	40	18	6	54	19	16	23	34	27	108	0

Таблица 1

a) (5 т.) Като се използва алгоритъма **A*** да се намери път от върха **A (Start)** до върха **M (Goal).**

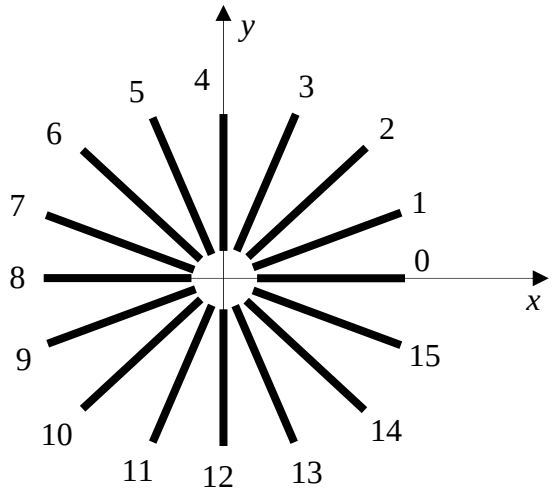
b) (2 т.) Като се използва алгоритъма за търсене в ширина да се намери път от върха **A (Start)** до върха **M (Goal).**

Задача 11. (4 т.) С псевдокод е представен алгоритъм на Брезенхам за растеризиране на отсечки с конкретни наклони.

```
draw_line( XA, YA, XB, YB )
    ΔX ← XB–XA
    ΔY ← YB–YA
    ε ← 0
    Δε ← |ΔY/ΔX|
    Y ← 0
    for X from XA to XB
        plot( X, Y )
        ε ← ε+Δε
        if ε≥0.5 then Y ← Y–1, ε ← ε+1
    endfor
end
```

Заградете с кръгче номерата на всяка от 16-те отсечки, която може да бъде растеризирана с представения алгоритъм при $X_A < X_B$.

Забележка: Отсечки с номера **2**, **6**, **10** и **14** са под ъгъл **±45°**.

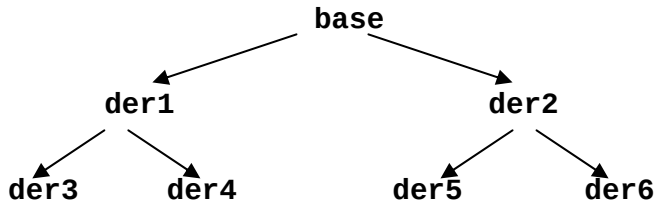


Задача 12. (4 т.) Допълнете кода на функцията **differ** така, че да проверява дали редицата от цели числа $a_0, a_1, \ldots a_{n-1}$ се състои от различни елементи.

```
bool differ ( int n, int a[ ] ) {

     int i, j ;

     ......................

     for ( ..................................... ; i++ )

         { ....................................

           for ( ..................................... ; j++ )

              { ....................................

                ....................................

               }

            ....................................

          }

     ....................................

     return  ..................................... ;  }
```

Задача 13. (3 т.) Дефинирайте рекурсивна функция **print_backward**, която извежда на екрана редица от цели числа, в техния обратен ред.

Задача 14. (5 т.) За йерархията



`class base`
`{ private: int b1;`
`  protected: int b2;`
`  public: int b3();`
`};`
	`class der1: public base`
	`{ private: int d11;`
	`  protected: int d12;`
	`  public: int d13();`
	`};`
		`class der2: protected base`
		`{ private: int d21;`
		`  protected: int d22;`
		`  public: int d23();`
		`};`

`class der3: der1`
`{ private: int d31;`
`  protected: int d32;`
`  public: int d33();`
`};`
	`class der4: protected der1`
	`{ private: int d41;`
	`  protected: int d42;`
	`  public: int d43();`
	`};`	

`class der5: public der2`
`{ private: int d51;`
`  protected: int d52;`
`  public: int d53();`
`};`
	`class der6: protected der2`
	`{ private: int d61;`
	`  protected: int d62;`
	`  public: int d63();`
	`};`	

определете възможностите за достъп на обектите: **b, d1, d2, d3, d4, d5** и **d6** до компонентите на класовете, ако:

```
base b;
der1 d1;   der2 d2;
der3 d3;   der4 d4;
der5 d5;   der6 d6;
```

Задача 15. Разгледайте програмата:

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

a) (5 т.) Намерете и обяснете грешките в процедурата main на горната програма.

b) (5 т.) Кои връзки в нея се разрешават статично и кои динамично?

c) (5 т.) Какъв е резултатът от изпълнението на програмата след отстраняване (задраскване) на неправилните обръщения към виртуланите функции?

Следният шаблон на клас реализира свързан списък, представен с две връзки:

фиг. 3

```
   template <class T>
struct elem_link2
{T inf;                                                          // елемент
 elem_link2<T> *pred,                                  // указател към предходния елемент
               *succ;                                   // указател към следващия елемент
};

template <class T>
class DLList
{public:
  DLList();                                                      // конструктор
  ~DLList( );                                                     // деструктор
  DLList(DLList const&);                                 // конструктор за присвояване
  DLList& operator=(DLList const &);                              // оператор =
  void print() const;                              // извежда списък, започвайки от началото му
  void print_reverse( ) const;                         // извежда списък, започвайки от края му
  void IterStart(elem_link2<T>* = NULL);                  // инициализира итератора CurrentS
  void IterEnd(elem_link2<T>* = NULL);                    // инициализира итератора CurrentE
  elem_link2<T>* IterSucc( );                 // връща текущата стойност на итертора CurrentS,
                                             // след което го премества в следващата му позиция
  elem_link2<T>* IterPred( );                 // връща текущата стойност на итертора CurrentE,
                                             // след което го премества в следващата му позиция
  void ToEnd(T const & x);                                // включва x в края на свързан списък
  void BeforeFirst(T const& x);              // включва x пред първия елемент на свързан списък
  void DeleteElem(elem_link2<T> * p, T & x);        // изключва елемента, сочен от указателя p
                                                                // и го запомня в параметъра x
  int length() const;                          // намира броя на елементите на двусвързан списък
private:
 elem_link2<T> *Start,                                          // указател към началото
               *End,                                             // указател към края
               *CurrentS,                                        // итератор към началото
               *CurrentE;                                           // итератор към края
 void DeleteList();
 void CopyList(DLList const &);
};
```

Задача 16. (2 т.) Реализирайте член-функциите на каноничното представяне на шаблона на класа **DLList** за дефиницията от фиг. 3.

Задача 17. (2 т.) Реализирайте член-функцията **`void IterStart(elem_link2<T>* = NULL);`** така че да инициализира итератора **`CurrentS`** за дефиницията от фиг. 3.

Задача 18. (2 т.). Реализирайте член-функцията **`elem_link2<T>* IterSucc( );`** така че да връща текущата стойност на итертора **`CurrentS`**, след което да го премества в следващата позиция. Използвайте дефиницията от фиг. 3.

Задача 19. (4 т.) Реализирайте член-функцията **`void DeleteElem(elem_link2<T> * p,T&x);`** така че да изключва елемента, сочен от указателя **`p`** и го запомня в параметъра **`x`**. Използвайте дефиницията от фиг. 3.

Задача 20. (4 т.). Реализирайте външна функция, която за двусвързан списък от символи проверява дали е палиндром. Използвайте дефиницията от фиг. 3.

Задача 21. (6 т.). Свързаният списък **l1** с елементи $a_1, a_2, \ldots, a_n$, представен с две връзки, е **<** от свързания списък **l2** с елементи $b_1, b_2, \ldots, b_m$, също представен с две връзки, ако **n <= m** и $a_n < b_m, a_{n-1} < b_{m-1}, \ldots, a_1 < b_{m-n+1}$. Предефинирайте оператора **<** за сравняване за **<** на два двусвързани списъка. Използвайте дефиницията от фиг. 3.