СОФИЙСКИ УНИВЕРСИТЕТ “СВ. КЛИМЕНТ ОХРИДСКИ” | **ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА**

ДЪРЖАВЕН ИЗПИТ
ЗА ПОЛУЧАВАНЕ НА ОКС “БАКАЛАВЪР ПО КОМПЮТЪРНИ НАУКИ”

ЧАСТ I (ПРАКТИЧЕСКИ ЗАДАЧИ)
15.07.2010 г.

Време за работа – 3 часа

Драги абсолвенти,
Попълнете факултетния си номер на всички страници!
Решението на всяка от задачите се разполага само в мястото от края на условието на тази задача до началото на условието на следващата задача. Могат да се използват и двете стани на листата

Изпитната комисия ви пожелава успешна работа.

Задача 1. (12 точки) Да се провери кои от следващите езици над азбуката X = {0,1} са едни и същи и кои са различни:

1. L_1 се представя чрез регулярния израз $1\ (0 + 1)^*\ 0$
2. L_2 се разпознава от крайния недетерминиран автомат

$$A = <\{q_0, q_1, q_2, q_3\}, \{0,1\}, q_0, \delta, \{q_3\}>$$

с функция на преходите δ, представена чрез таблицата:

q	0	1
q_0	$\{q_1\}$	$\{q_2\}$
q_1	$\{q_1,q_3\}$	$\{q_1\}$
q_2	$\{q_2\}$	$\{q_2,q_3\}$
q_3	$\varnothing$	$\varnothing$

3. L_3 се разпознава от крайния детерминиран автомат

$$B = <\{q_0, q_1, q_2\}, \{0,1\}, q_0, \delta, \{q_0, q_1, q_2\}>$$

с функция на преходите δ, представена чрез таблицата:

q	0	1
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_0	—

4. L_4 се разпознава от крайния детерминиран автомат

$$C = <\{q_0, q_1, q_2, q_3, q_4\}, \{0,1\}, q_0, \delta, \{q_2, q_4\}>$$

с функция на преходите δ, представена чрез таблицата:

q	0	1
q_0	q_1	q_3

q_1	q_2	q_1
q_2	q_2	q_1
q_3	q_3	q_4
q_4	q_3	q_4

За да покажете, че два езика са различни посочете дума, която е от единия език, но не е от другия, а за да покажете, че два езика съвпадат сравнете крайните детерминирани амтомати, които ги разпознават.

Задача 2. (12 точки) Текстов файл с име comprocB съдържа зададената по-долу последователност от команди на bash за Linux. Напишете вдясно какво ще бъде изведено на стандартния изход след стартиране на файла с команден ред: **bash comprocB b1 b2 b3,** ако на стандартния вход бъде подадена следната последователност от символи: **b2**

```
if  test –z $5
then echo $1
        for var
        do  echo  $var >> fniz
        done
else echo $2
       while true
        do echo LOOP
         break
         done
fi
cat fniz
read  string
until  cat fniz | grep $string
do
    set  $2  b1
    echo  `grep  $2  fniz `
    echo  END
    exit
done
set   $3  $1  1
echo OK $3
echo  ` grep  $1  fniz `
exit
echo  OK
```

Задача 3. (12 точки) Да се напише функция, която по зададен масив от низове намира най-големия брой низове от масива, които са *анаграми* помежду си. Един низ е анаграма на друг, ако е съставен от същите символи, но в разбъркан ред, като това означава, че за да са анаграми два низа трябва всеки от символите на първия да се среща точно толкова пъти и във втория. Например низовете asdff и fsdaf са анаграми един на друг, докато низовете asdff и aafsd не са.

Примери:
Вход: `string ginrts ringst strong spong shpong pongs`
Изход: 3

Думите могат да бъдар разделени на следните групи, елементите на които са анаграми помежду си:

```
string ginrts ringst
strong
shpong
spong pongs
```

Задача 4. (15 точки**)** Класът DLList, представя списък с две връзки, съдържащ числа с плаваща запетая. Дефиницията на класа е следната:

```
struct node {
  double data;
  node* next;
  node* prev;
};

class DLList {
  private:
    node* start;         // указател към началото на списъка
    node* end;           // указател към края на списъка
    node* forwardIter;   // итератор за обхождане напред
    node* backwardIter;  // итератор за обхождане назад
  public:
    // функции от голямата четворка
    DLList();
    DLList(DLList const&);
    DLList& operator=(DLList const&);
    ~DLList();
    // вмъква числото x пред клетката p
    void insertBefore(node* p, double x);
    // вмъква числото x след клетката p
    void insertAfter(node* p, double x);
    // премахва клетката p от списъка
    void insertAfter(node* p, double x);
    // установява forwardIter да сочи към началото на списъка
    // или към клетката p, ако ѝ е зададена ненулева стойност
    void startForward(node* p=NULL);
    // установява backwardIter да сочи към началото на списъка
    // или към клетката p, ако ѝ е зададена ненулева стойност
    void startBackward(node* p=NULL);
    // премества forwardIter напред и връща старата му стойност
    node* nextForward();
    // премества backwardIter напред и връща старата му стойност
    node* nextBackward();
};
```

а) (3 т.) Да се дефинират член-функциите от голямата четворка;

б) (3 т.) Да се дефинират член-функциите insertBefore и insertAfter;

в) (2 т.) Да се дефинира член-функцията deleteElem;

г) (3 т.) Да се дефинира функцията bool mirror(DLList& dl1, DLList& dl2), която проверява дали списъкът dl1 е огледален образ на dl2, т.е. дали dl1 се състои от елементите на dl2 в обратен ред;

д) (4 т.) Да се дефинира функцията node* split(DLList& dl, double x), която с едновременно обхождане от началото и края нарежда елементите на списъка dl в две области така, че в началото на списъка да се окажат само числа, по-малки от x, а в края на списъка — числа, по-големи от x. За целта да се използва следния алгоритъм: списъкът dl се обхожда от двата края до момента, в който бъдат достигнати два елемента a (от началото) и b (от края), такива че a > x > b. При достигане на такива елементи те да бъдат разменени и обхождането да продължи със същата стратегия. Като резултат да бъде върната границата между двете области, т.е. указател към клетката, съдържаща последния по ред елемент в dl, който е по-малък от x.

Задача 5. (10 точки) Намерете асимптотичната сложност по време на алгоритъма, представен със следния фрагмент:

```
int main() { r(1, n, n*n); }

void r(int a, int b, int c) {
   int k;
   if(a + b + c > a + b + 1) {
      for(k = 1; k < a+b+c; k = (k<<2) – 1)
      {
         if(k%11 == 0) break;
         else r(a, b, c-1);
      }
      for(k = 1; k <a+b+c; k <<= a+b+c)
         r(a,b,c-1);
   }
}
```

Задача 6. (12 точки) Нека е даден списък от естествени числа *L*, който съдържа цифри (естествени числа в интервала [0, 9]). Напишете функция (find-max L), която намира най-голямото число, което може да се образува от цифрите в L. Ако L е празен, функцията find-max трябва да връща нула.

Пример:

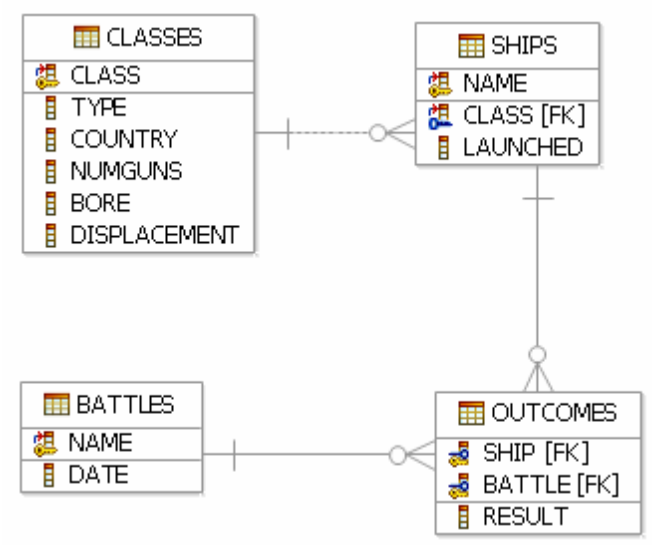
```
(find-max '()) → 0
(find-max '(0 0 0)) → 0
(find-max '(1 1 9 8 9 3 4 6 7 0 0)) → 99876431100
```

Задача 7. (?? точки) С метода на резолюцията докажете, че от φ_1, φ_2 и φ_3 следва ψ, където

φ_1: $\forall X \forall Y(\ p(X, Y) \Rightarrow q(X, Y)\ )$
φ_2: $\forall X \exists Y(\ q(X, X) \,\&\, r(X, Y) \Rightarrow s(Y, X)\ )$
φ_3: $\exists X \forall Y(\ p(X, X) \,\&\, r(X, Y)\ )$
ψ: $\exists X \exists Y(\ p(X, X) \,\&\, s(Y, X)\ )$.

Задача 8. (10 точки) Дадено е пространство от състояния $V = \{v_0, v_1, v_2, \ldots v_n\}$. Състоянието v_0 е начално, а v_n – крайно. За всяко състояние е дефинирана функцията f(v), която пресмята очакваната цена на пътя от това състояние до крайното, ($f(v_n) = 0$). Има и функция на преходите p(v), която за всяко състояние предоставя множество от състоянията, които са негови непосредствени наследници. Приемаме, че цената на всеки преход между две състояния е еднаква. Дадено е множество $Bad \subset V$. Да се реализира алгоритъм за намиране на път с минимална цена между началното и крайното състояние, който не минава през състояния от множеството Bad. За реализацията използвайте константно зададени в програмата входни данни. Приемете, че имате дефинирани функциите f и p.

Задача 9. Дадена е базата от данни Ships, в която 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*).



Таблицата ***Classes*** съдържа информация за класовете кораби:

- *class* – име на класа, първичен ключ;
- *type* – тип (‘bb’ за бойни кораби и ‘bc’ за бойни крайцери);
- *country* – държавата, която строи такива кораби;
- *numGuns* – броя на основните оръдия;
- *bore* – калибъра им (диаметърът на отвора на оръдието в инчове);
- *displacement* – водоизместимост (в тонове).

Таблицата ***Ships*** съдържа информация за корабите:

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

Таблицата ***Battles*** съхранява информация за битките:

- *name* – име на битката, първичен ключ;
- *date* – дата 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* – име на кораба;
- *battle* – име на битката;
- *result* – резултат (потънал-‘sunk’, повреден – ‘damaged’, победил – ‘ok’).

1. Посочете заявката, която извежда всички държави, които имат както класове с по-малко от 9 оръдия (*numguns*), така и класове с над 12 оръдия:

```
а)
SELECT DISTINCT COUNTRY
FROM CLASSES
WHERE NUMGUNS<9 AND NUMGUNS>12;

б)
SELECT DISTINCT C1.COUNTRY
FROM CLASSES C1
JOIN CLASSES C2 ON C1.COUNTRY=C2.COUNTRY
WHERE C1.NUMGUNS<9 AND C2.NUMGUNS>12;

в)
SELECT COUNTRY
FROM CLASSES
WHERE NUMGUNS<9
UNION
SELECT COUNTRY
FROM CLASSES
WHERE NUMGUNS>12;

г)
SELECT DISTINCT COUNTRY
FROM CLASSES
WHERE NUMGUNS<9 AND COUNTRY =
```

```
(SELECT COUNTRY
FROM CLASSES
WHERE NUMGUNS>12);
```

2. Посочете заявката, която за всяка държава, участвала в не повече от 4 битки, извежда името ѝ и броя битки, в които е участвала. Ако дадена държава няма нито един кораб или не е участвала в нито една битка, за нея да извежда 0.

```
а)
SELECT COUNTRY, COUNT(DISTINCT BATTLE)
FROM CLASSES C, SHIPS S, OUTCOMES O
WHERE C.CLASS=S.CLASS AND S.NAME=O.SHIP
GROUP BY COUNTRY
HAVING COUNT(DISTINCT O.BATTLE)<4;

б)
SELECT COUNTRY, COUNT(O.BATTLE) AS NUM_BATTLES
FROM CLASSES C
LEFT JOIN SHIPS S ON C.CLASS=S.CLASS
LEFT JOIN OUTCOMES O ON S.NAME=O.SHIP
GROUP BY COUNTRY
HAVING COUNT(O.BATTLE)<4;

в)
SELECT COUNTRY, COUNT(DISTINCT BATTLE)
FROM OUTCOMES
JOIN SHIPS ON NAME=SHIP
RIGHT JOIN CLASSES ON CLASSES.CLASS=SHIPS.CLASS
GROUP BY COUNTRY
HAVING COUNT(DISTINCT OUTCOMES.BATTLE)<=3;

г)
SELECT C.COUNTRY, COUNT(O.BATTLE)
FROM CLASSES AS C
INNER JOIN SHIPS AS S ON C.CLASS=S.CLASS
LEFT OUTER JOIN OUTCOMES AS O ON S.NAME=O.SHIP
WHERE COUNT(O.BATTLE)<=3;
```