

**СОФИЙСКИ УНИВЕРСИТЕТ  
„СВ. КЛИМЕНТ ОХРИДСКИ“**



**ФАКУЛТЕТ ПО МАТЕМАТИКА  
И ИНФОРМАТИКА**

**ДЪРЖАВЕН ИЗПИТ  
ЗА ПОЛУЧАВАНЕ НА ОКС “БАКАЛАВЪР ПО ИНФОРМАТИКА”**

**ЧАСТ I (ПРАКТИЧЕСКИ ЗАДАЧИ)  
15.07.2014 г.**

**Време за работа – 3 часа**

*Драги абсолвенти:*

- Попълнете факултетния си номер в горния десен ъгъл на всички страници;
- Пишете само на предоставените листове без да ги разкопчавате;
- За всяка от задачите, беловата с решението може да е само на листите, на които е изписано условието на съответната задача, или на празна страница след условието. При необходимост пренасяте решението на подпечатан нов лист със заглавен текст „Задача N, стр. M, ф.н. F“, където M ( $M \geq 1$ ) е поредния номер допълнителен лист за задача N, а F е вашият факултетен номер.

*Изпитната комисия ви пожелава успешна работа!*

**Задача 1.** (10 т.) Даден е неориентиран граф  $G(V, E)$ , в който всеки връх има степен поне  $d \geq 2$ .

а) Докажете, че в  $G$  има цикъл.

б) Докажете по-силно твърдение: в  $G$  има цикъл с дължина поне  $d + 1$ .

*Упътване: разгледайте произволен най-дълъг път  $p$  в  $G$ , в който няма повтаряне на върхове. Разгледайте всички съседи на някой от крайните върхове на  $p$ .*

Задача 2. (10 т.) Даден е недетерминираният краен автомат

$$A = \langle \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}, \{a, b, c\}, q_0, \delta, \{q_4, q_5\} \rangle$$

с функция на преходите  $\delta$ , определена както следва:

$\delta$	$a$	$b$	$c$
$q_0$	$\{q_1, q_3, q_5\}$	$\{q_5\}$	$\emptyset$
$q_1$	$\emptyset$	$\emptyset$	$\{q_2\}$
$q_2$	$\emptyset$	$\{q_4\}$	$\emptyset$
$q_3$	$\emptyset$	$\{q_3, q_4\}$	$\emptyset$
$q_4$	$\emptyset$	$\emptyset$	$\emptyset$
$q_5$	$\emptyset$	$\emptyset$	$\{q_6\}$
$q_6$	$\emptyset$	$\emptyset$	$\{q_7\}$
$q_7$	$\{q_7\}$	$\{q_5\}$	$\emptyset$

Да се построи детерминиран краен автомат  $A'$ , еквивалентен на  $A$ .

**Задача 3.** (10 т.) Текстов файл с име `comproc1` съдържа зададената по-долу последователност от команди на `bash` за Linux. Напишете внятно какво ще бъде изведено на стандартния изход след стартиране на файла с команден ред

```
bash comporc1 ab cd ef
```

ако на стандартния вход бъде подадена следната последователност от символи: 1 2

```
count=1
for i in 5 1 4 2
do for j
    do if test $i -ge $#
        then count=`expr $count \* $i`
            echo $count $j >> f1
        else while true
            do echo $*
                break 3
            done
        fi
    done
done
read k1 k2
while cat f1 | grep $k2
do set $k1 $count
    shift
    echo $2
    echo $1 $i
    exit
done
echo FIN
```

**Задача 4.** (10 т.) *Задачата да се реши на езика C++.*

Даден е символен низ с дължина  $n$ , съдържащ само малки латински букви. Напишете функция `void check(const char* str)`, която проверява дали съществува позиция в низа, такава че, ако се започне четене от тази позиция и четенето продължи до края и след това (при необходимост) циклично се прехвърли в началото на низа, може да се прочете палиндром с дължина равна на  $n$  (дължината на входния низ). Ако такава позиция съществува, функцията да извежда на екрана нейния номер и намереният палиндром, ограден в кръгли скоби. Ако такава позиция не съществува, функцията да извежда на екрана текста "NO".

Примери:

Вход: orrobborrow    Изход: 5 (borroworrob)

Вход: borroworrob    Изход: 0 (borroworrob)

Вход: booo            Изход: NO

**Задача 5.** (10 т.) *Задачата да се реши на езика C++.*

Дадена е квадратна матрица от цели числа с размери  $10 \times 10$ , която описва лабиринт. Стойност 0 в дадена клетка означава „стена“, а стойност 1 означава „проходима клетка“. Даден е символен низ, съдържащ само буквите E, W, N и S, които указват едностъпкови придвижвания в съответните географски посоки: N – нагоре, E – надясно, S – надолу, W – наляво.

Да се напише функция `walk`, която получава матрица и символен низ от вида, определен по-горе и проверява дали символният низ задава валиден път започващ от някоя проходима клетка на лабиринта, състоящ се само от проходими клетки и завършващ в долния десен ъгъл на лабиринта. Функцията да връща булева стойност – *истина*, ако такава клетка има и даденият низ задава валиден път и *лъжа* в противен случай.

**Задача 6.** (10 т.) *Задачата да се реши на езика C++ или Java. В началото на вашето решение посочете кой език сте избрали.*

А) Да се дефинира структура `ChessPosition`, описваща коректна позиция на фигура върху шахматна дъска (координатите на позицията са от 'А' до 'Н' по едното измерение и от 1 до 8 по другото).

Да се дефинира абстрактен клас (или интерфейс) `ChessPiece`, описващ шахматна фигура със следните операции:

- `ChessPosition getPosition()` – Връща позицията на фигурата на дъската;
- [подходящ тип] `allowedMoves()` – Връща списък (колекция) с всички възможни позиции, до които дадена фигура може да достигне с един ход;
- [булев тип] `captures(ChessPosition pos)` – Проверява дали фигурата “владее” позицията `pos`, подадена като параметър, т.е. дали позицията е в списъка с възможните ходове на фигурата. Булевият тип да бъде булевият тип в езика, който сте избрали (напр. `bool`, ако пишете на C++).

Б) Да се дефинират класовете `Rook` и `Knight` – наследници на `ChessPiece`, описващи съответно шахматните фигури топ и кон.

В) „Стабилна конфигурация“ наричаме такава подредба на фигурите върху дъската, при която никоя фигура да не е върху позволен ход на друга фигура (т.е. никои две фигури да не се „бият“). Да се дефинира функцията `allMoves` ([подходящ тип] `pieces[, ...]`), която за списъка (колекцията) `pieces`, съдържащ произволен брой разнородни шахматни фигури, отпечатва на конзолата всеки възможен ход на фигура от `pieces` такъв, че след изпълнението му списъкът с фигури да описва стабилна конфигурация. Информацията за ходовете да съдържа типа на фигурата, старата позиция и новата позиция, например:

`Rook A1 -> B1`

`Knight B3 -> A5`

*Забележка: Реализирайте всички конструктори и други операции, които смятате, че са необходими на съответните класове.*

**Задача 7.** (10 т.) Задачата да се реши на езика C++ или Java. В началото на вашето решение посочете кой език сте избрали.

Едносвързан цикличен списък от цели числа се описва с референция (указател) към циклична верига от двойни клетки, представени по следния начин:

C++

```
struct Node {  
    Node *next;  
    int data;  
};
```

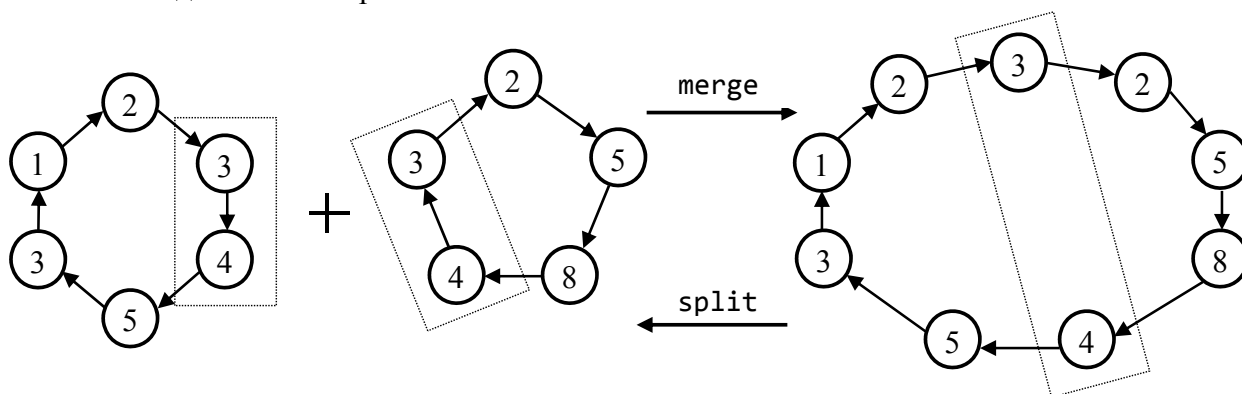
Java

```
public class Node {  
    public Node next;  
    public int data;  
}
```

А) За даден цикличен списък L казваме, че числото M предхожда числото N, ако в списъка L има кутия A, която съдържа M, а A.next сочи към кутия, която съдържа N.

Да се реализира (статична) функция `precedes`, която по даден списък L и две числа M и N проверява дали M предхожда N в L и ако е така, връща референция (указател) към кутията A, която съдържа M.

Б) Ако са дадени два списъка L1 и L2, така че M предхожда N в L1 и N предхожда M в L2, тогава двата списъка могат да се слоят в един списък L, както е показано на диаграмата долу, така че в L остава само по един екземпляр на числата M и N.



Да се реализира (статична) функция `merge`, която слива два списъка L1 и L2, ако това е възможно. В случай, че сливането може да стане по няколко различни начина, да се избере такъв, за който сумата на числата M и N е максимална.

В) Да се реализира (статична) функция `split`, която по даден списък L и числа M и N, ако е възможно, разделя списък L на два списъка L1 и L2, така че M предхожда N в L1 и N предхожда M в L2, както е показано на диаграмата горе. Ако разделянето може да стане по повече от един начин, да се избере такъв, при който разликата между дължините на получените списъци L1 и L2 е минимална.

**Задача 8.** (10 т.) *Задачата да се реши на езика Scheme или Haskell. В началото на вашето решение посочете кой език сте избрали.*

А) Напишете функция `totalMin`, която за списък от едноместни числови функции връща тази функция  $f$  от списъка, за която  $f(0)$  е минимално.

Б) Напишете функция `chainMinCompositions`, която получава като аргумент едноместна числова функция  $f$  и генерира безкрайния поток (за Хаскел – безкрайния списък)  $F_0, F_1, F_2, \dots$ , където:

$$F_0 = id$$

$$F_1 = f$$

$$F_i = F_{i-1} \circ F_{i-2}, \text{ ако } i > 1 \text{ и } F_{i-1}(j) \neq F_{i-2}(j), \text{ за някое цяло число } j \in [0, i]$$

$$F_i = totalMin \{F_0, F_1, \dots, F_{i-1}\}, \text{ ако } i > 1 \text{ и } F_{i-1}(j) = F_{i-2}(j), \text{ за всяко цяло число } j \in [0, i]$$

*Забележка: с  $id$  е означена функцията „идентитет“, като  $id(x) = x$  за произволно  $x$ , а с  $f \circ g$  е означена композицията на функциите на  $f$  и  $g$ , като  $(f \circ g)(x) = f(g(x))$ .*