СОФИЙСКИ УНИВЕРСИТЕТ
“СВ. КЛИМЕНТ ОХРИДСКИ”

ФАКУЛТЕТ ПО МАТЕМАТИКА
И ИНФОРМАТИКА

ДЪРЖАВЕН ИЗПИТ

ЗА ПОЛУЧАВАНЕ НА ОКС “БАКАЛАВЪР ПО КОМПЮТЪРНИ НАУКИ”

ЧАСТ I (ПРАКТИЧЕСКИ ЗАДАЧИ)

Драги абсолвенти:

- Попълнете факултетния си номер в горния десен ъгъл на всички листове.
- Пишете само на предоставените листове, без да ги разкопчавате.
- Решението на една задача трябва да бъде на същия лист, на който е и нейното условие (т.е. може да пишете отпред и отзад на листа със задачата, но не и на лист на друга задача).
- Ако имате нужда от допълнителен лист, можете да поискате от квесторите.
- На един лист не може да има едновременно и чернова, и белова.
- Черновите трябва да се маркират, като най-отгоре на листа напишете “ЧЕРНОВА”.
- Ако решението на една задача не се побира на нейния лист, трябва да поискате нов бял лист от квесторите. Той трябва да се защипе с телбод към листа със задачата.
- Всеки от допълнителните листове (белова или чернова) трябва да се надпише най-отгоре с вашия факултетен номер.
- Черновите също се предават и се защипват в края на работата.
- Времето за работа по изпита е 3 часа.

Изпитната комисия ви пожелава успешна работа!

Задача 1. Задачата да се реши на езика C++.

1) Нека е дефиниран масивът
`int arr[] = { 1, 2, 3 };`
Срещу всеки от изразите да се посочи каква ще бъде неговата оценка.

```
arr[1] == *(arr+2) _____________________
arr == &arr[0]     _____________________
(arr+1) == &arr[1] _____________________
*arr == arr[0]     _____________________
```

2) Нека е дадена следната дефиниция:

```
void mystery(const char* str)
{
  while (*str && *(str+1)) {
    std::cout << *str;
    str += 2;
  }
}
```

Да се посочи какво ще изведе на екрана обръщението:

```
mystery("abcdef");
```

_____

3) Нека са дадени следните дефиниции:

```
char s1[] = "Hello";
char s2[] = "world!";
char result[80];
```

Да се довърши програмният фрагмент, така че след изпълнението му в `result` да се съхрани коректното представяне на низа `"Hello world!"`. На празните места трябва да се попълнят имената на подходящи стандартни функции за работа с низове.

```
_____________________(result, s1);
_____________________(result, " ");
_____________________(result, s2);
```

4) Да се довърши кодът на рекурсивните функции, така че `f` да проверява дали символният низ, сочен от `word`, се съдържа като подниз в `text`. За определеност считаме, че празният низ се съдържа във всеки друг.

```
bool g(const char* text, const char* word)
{
  if (!*word) return true;
  if (!*text) return _______________;
  if (*word != *text) return false;
  return g(_______________, _______________);
}

bool f(const char* text, const char* word)
{
  if (!*word) return _______________;
  if (!*text) return false;
  return g(_______________, _______________) ||
         f(_______________, _______________);
}
```

5) Да се посочи какво ще изведе на екрана даденият по-долу фрагмент:

```
char arr[3][3] = { 'a', 'b', 'c',
                   'd', 'e', 'f',
                   'g', 'h', 'i' };
for (int i = 0; i < 3; ++i)
  std::cout << arr[2-i][i];
```

_____

6) Да се посочи какво ще изведе на екрана даденият по-долу фрагмент:

```
double var = 5 / 2;
std::cout << var;
```

_____

Критерии за оценяване

- Точки се дават само за напълно коректно посочени отговори.
- В подточките, в които се изисква да се посочи какво ще се изведе, точки се дават само ако отговорът напълно съвпада с това, което извежда съответният код. В противен случай се дават 0 т.
- В задачите за посочване на стойност на израз, ако отговорът не съвпада напълно с коректното решение се дават нула точки.
- В задачите за посочване на стойност на израз, ако вместо булевите литерали true/false се посочат числата 1/0, точките се намаляват наполовина.
- В задачата за довършване на кода на функцията, ако написаното не е синтактично или логически коректно или е различно от коректния отговор, се дават 0 т.
- Сумата от точките се закръгля до цяло число.

Максималната оценка за всяка подточка е както следва:

- Подточка 1: 2 точки (по 0.5 точки за всеки напълно коректен отговор)
- Подточка 2: 1 точка
- Подточка 3: 1 точка (0.5 точки, ако коректно е посочен `strcpy` за първата функция; 0.5 точки, ако коректно е посочена `strcat` за вторите две извиквания).
- Подточка 4: 4 точки (по 0.5 точки за всеки напълно коректен отговор).
- Подточка 5: 1 точка
- Подточка 6: 1 точка (отговорът се счита за коректен, независимо дали е посочен с десетична точка; например 2, 2.0 или 2,0 са коректни отговори).

Примерно решение:

1)

```
arr[1] == *(arr+2) --> false
arr == &arr[0] --> true
(arr+1) == &arr[1] --> true
*arr == arr[0] --> true
```

2)

```
ace
```

3)

```
strcpy(result, s1);
strcat(result, " ");
strcat(result, s2);
```

4)

```
bool g(const char* text, const char* word)
{
  if (!*word) return true;
  if (!*text) return false;
  if (*word != *text) return false;
  return g(text + 1, word + 1);
}

bool f(const char* text, const char* word)
{
  if (!*word) return true;
  if (!*text) return false;
  return g(text, word) ||
         f(text + 1, word);
}
```

5)

```
gec
```

6)

```
възможни отговори: 2, 2.0, 2,0 и т.н.
```

Задача 2. Задачата да се реши на езика C++.

1) Освен конструктора по подразбиране (default constructor), кои други функции влизат в "голямата четворка" (функциите от т.нар. "rule-of-3")? Да се попълнят имената им в полетата долу:

_____
_____
_____

2) Нека е дадена дефиницията:

```
class foo {
public:
  virtual void f() {};
  void g() {};
};
```

Срещу всеки от редовете, които извикват `f` или `g`, да се запише "статично" или "динамично" според вида свързване, който ще се използва за тях.

```
foo obj;
foo& ref = obj;
obj.f();  ____________________
obj.g();  ____________________
ref.f();  ____________________
ref.g();  ____________________
```

3) Нека са дадени следните дефиниции:

```
class base {
public:  int a;
private: int b;
};
class derived : protected base { };
```

Да се посочи каква ще бъде видимостта на променливите `a` и `b` в класа `derived` – `public`, `protected` или `private`.

- Видимост на `a`: _____
- Видимост на `b`: _____

4) Нека класът `X` е абстрактен. Срещу всяко от твърденията да се посочи "да" или "не" според това дали е вярно:

- Могат да се създават обекти от тип `X`: ____
- Могат да се създават референции (reference) към обекти от тип `X`: ____

5) Нека е дадена следната дефиниция:

```
struct s {
public:
  static int var;
  s() { var = 5; }
};
int s::var = 0;
```

Да се посочи какво ще изведе следният фрагмент:

```
std::cout << '(' << s::var << ')';
s obj1;
obj1.var = 10;
s obj2;
std::cout << '-' << s::var << '-';
```

_____

6) Да се допълни дефиницията на класа `test`, така че функцията `f` да бъде чиста виртуална (pure-virtual) и класът да може коректно да се използва като основа на полиморфна йерархия.

```
class test {
public:
  _____________ void f() _____________;
  ____________________________________
};
```

7) Да се допълни дефиницията на шаблона `Array`, така че функцията `test` да се компилира коректно и да извежда на стандартния изход `55`.

```
_____________ <_____________>
class Array {
  static const size_t size = 10;
  T data[size];
public:
  _____________ at(size_t index) {
    if (index _____________________)
      throw std::out_of_range("error");
    return data[index];
  }
};

void test() {
  Array<int> a;
  a.at(0) = 5;
  std::cout << a.at(0);
  Array<Array<int>> b;
  b.at(0) = a;
  std::cout << b.at(0).at(0);
}
```

Критерии за оценяване

- Точки се дават само за напълно коректно посочени отговори.
- В подточките, в които се изисква да се посочи какво ще се изведе, точки се дават само ако отговорът напълно съвпада с това, което извежда съответният код. В противен случай се дават 0 т.
- В задачите за посочване на стойност на израз, ако отговорът не съвпада напълно с коректното решение, се дават нула точки.
- В задачите за посочване на стойност на израз, ако вместо булевите литерали `true` и `false` се посочат числата 1 и 0, точките се намаляват наполовина.
- В задачата за довършване на кода на функцията, ако написаното не е синтактично или логически коректно или е различно от коректния отговор, се дават 0 т.
- Сумата от точките се закръгля до цяло число.

Максималната оценка за всяка подточка е както следва:

1. 1 точка (1/3 точки за коректен отговор, −1/3 точки за грешно посочена функция, минимална оценка: 0 точки).
2. 2 точки (по 0.5 за коректен отговор).
3. 1 точка (по 0.5 за коректен отговор).
4. 1 точка (по 0.5 за коректен отговор).
5. 1 точка (по 0.5 съответно ако е посочено извеждането на 0 и на 5; Ако в отговора не са включени скобите и тиретата, оценката се намалява с 0.2 точки).
6. 2 точки (по 0.5 съответно за `virtual` и за `= 0` във функцията `f`; 1 точка за виртуален деструктор).
7. 2 точки (по 0.5 за всяко коректно попълнено място).

Примерно решение:

1)

- “копиращ конструктор” или “конструктор за копиране” или “copy constructor”
- “копиращо присвояване” или “операция/оператор за присвояване” или “copy assignment” или “assignment operator” или “`operator=`”
- “деструктор” или “destructor”

2)

```
foo obj;
foo& ref = obj;
obj.f();  статично
obj.g();  статично
ref.f();  динамично
ref.g();  статично
```

3)

- Видимост на `a`: `protected`
- Видимост на `b`: `private` или “няма видимост в производния клас”

4)

- Могат да се създават обекти от тип `X`: не
- Могат да се създават референции (reference) към обекти от тип `X`: да

5)

```
(0)-5-
```

6)

```
class test {
public:
  virtual void f() = 0;
  virtual ~test() {}
};
```

7)

```
template <typename T>
class Array {
  static const size_t size = 10;
  T data[size];
public:
  T& at(size_t index) {
    if (index >= size)
      throw std::out_of_range("error");
    return data[index];
  }
};
```

Задача 3. Задачата да се реши на езика C++. Там, където се изисква да се посочи сложност, тя трябва да се изрази в термините на Θ нотацията. Забележка: $f(n) = \Theta(g(n))$, ако f е асимптотично ограничена отгоре и отдолу от g, например $3n^2 + n = \Theta(n^2)$, но $5n \neq \Theta(n^2) \neq 8n^3 + 2n^2$.

В тази задача едносвързан списък ще представяме чрез указател към първия му елемент. Елементите на списъка представяме чрез структури от вида:

```
struct node {
  unsigned data;
  node* next; // Указател към следващ елемент в списъка или nullptr, ако няма такъв.
  node(unsigned data) : data(data), next(nullptr) {}
};
```

1) Да да се довърши кода на функцията `copyEveryOther`, така че тя да връща нов списък, в който се копират само елементите на нечетни позиции в списъка, сочен от `lst`, запазвайки техния ред. Приемаме, че позициите са номерирани от 1.

```
node* copyEveryOther(node* lst) {
  if (!lst) return nullptr;
  node* result = new node(_________________);
  node* p = _________________;
  while ((lst = lst->next) && (_________________) ) {
      p->next = new node(_________________);
      p = _________________;
  }
  return result;
}
```

2) Каква е времевата сложност на алгоритъма от 1) за списък с дължина n?

_____

3) Каква е външната (auxiliary) пространствена сложност на алгоритъма от 1) за списък с дължина n? Във "външната сложност" НЕ СЕ включва пространството, консумирано от входните данни, нито това за изходните данни.

_____

4) Нека е дадено двоично наредено дърво. То използва стандартната организация, при която за всеки връх вляво от него има по-малки, а вдясно – по-големи елементи. В началото дървото е празно. След това последователно в него се добавят числата 5, 10, -5, 1 и 12. След това елементите на дървото се обхождат "ляво-корен-дясно" (inorder) и се извеждат на екрана. В дадените по-долу полета да се попълнят елементите в реда, в който ще бъдат изведени.

_____, _____, _____, _____, _____

5) Нека е дадено произволно двоично наредено дърво, съдържащо n елемента. Каква е времевата сложност на търсенето на елемент в него в най-лошия случай?

_____

6) Да се посочат времевите сложности на следните операции с едносвързан списък с дължина n, който ни е подаден чрез указател към първия му елемент:

Премахване на първия му елемент: _____

Премахване на последния му елемент: _____

Проверка дали в списъка се съдържа дадено число K: _____

Добавяне на число на втора позиция в списъка: _____

Критерии за оценяване

- Точки се дават само за напълно коректно посочени отговори.
- В задачите за посочване на сложност, ако е посочена некоректна сложност се дават 0 точки.
- В задачата за довършване на кода на функцията, ако написаното не е синтактично или логически коректно или е различно от коректния отговор, се дават 0 т.
- В задачата за обхождане на дърво, ако елементите не са подредени коректно се дават 0 точки.
- Сумата от точките се закръгля до цяло число.

По-конкретно, за отделните подточки:
1) Всяко напълно коректно попълнено празно място носи 0,4 т. (общо 2 т.)
2) Носи 1 точка.
3) Носи 1 точка.
4) Носи 1 точка.
5) Носи 1 точка.
6) Всяка напълно коректно посочена сложност носи 1 точка (общо 4 т.).

Примерно решение на задачата

1)

```
node* copyEveryOther(node* lst) {
  if (!lst) return nullptr;
  node* result = new node(lst->data);
  node* p = result;
  while ((lst = lst->next) && (lst = lst->next)) {
      p->next = new node(lst->data);
      p = p->next;
  }
  return result;
}
```

2) $\Theta(n)$

3) $\Theta(1)$

4) $-5, 1, 5, 10, 12$

5) $\Theta(n)$

6)

- Премахване на първия му елемент: $\Theta(1)$
- Премахване на последния му елемент: $\Theta(n)$
- Проверка дали в списъка се съдържа дадено число K: $\Theta(n)$
- Добавяне на число на втора позиция в списъка: $\Theta(1)$

Задача 4. Крайно кореново дърво, всеки връх на което съдържа цяло число и може да има произволен брой деца, се представя в Scheme като наредена двойка, състояща се от стойността на корена и списък от директни поддървета, а в Haskell — със следната рекурсивна структура:

```
data Tree = T { root :: Int, subtrees :: [Tree] } deriving Show
```

Листо наричаме дърво с единствен връх, **клонка** наричаме дърво, чиито директни поддървета са листа, а **пръчка** наричаме дърво, в което всеки връх има най-много едно дете. Казваме, че едно дърво се **подрязва**, ако от него се премахнат всички клонки, с изключение на корена, и че се **окастря**, ако всички пръчки в него, които са с повече от два върха и не са част от други пръчки, се скъсят откъм листата до дължина точно два върха. Да се попълнят празните полета по-долу, така че:

- функциите `leaf`, `twig` и `stick` да проверяват дали дърво е съответно листо, клонка или пръчка;
- функциите `trim` и `prune` да връщат съответно подрязано или окастрено копие на дървото, подадено им като параметър.

Упътване: *могат да се използват наготово `all`, `any`, `car`, `cdr`, `filter`, `foldl`, `foldr`, `head`, `null`, `null?`, `tail`, `length`, `map`, както и всички функции в R5RS за Scheme и в Prelude за Haskell.*

Haskell

```
leaf  _____
twig  _____
stick _____
      _____

trim (T x ts) = T x _____
                    _____

prune t@(T x []) = _____
prune t@(T x ts) = T x (if stick t
                        then _____
                        else _____
```

Пример:

```
twig (T 1 [T 2 [], T 3 []]) ⟶ True          stick (T 1 [T 2 [T 3 [T 4 []]]]) ⟶ True
tree = T 1 [T 2 [T 3 []], T 4 [T 5 [T 6 []]], T 7 [T 8 [], T 9 [T 10 [T 11 []]]]]
trim tree  ⟶ T 1 [T 4 [], T 7 [T 9 []]]
prune tree ⟶ T 1 [T 2 [T 3 []], T 4 [T 5 []], T 7 [T 8 [], T 9 [T 10 []]]]
```

Scheme

```
(define (leaf t)  _____)
(define (twig t)  _____)
(define (stick t) _____)
(define (trim t)
  (cons (car t) _____
                _____))
(define (prune t)
  (if (leaf t)  _____
      (cons (car t) (if (stick t)
                        _____
                        _____))))
```

Пример:

```
(twig '(1 (2) (3))) ⟶ #t                    (stick '(1 (2 (3 (4))))) ⟶ #t
(define tree '(1 (2 (3)) (4 (5 (6))) (7 (8) (9 (10 (11))))))
(trim tree)  ⟶ (1 (4) (7 (9)))
(prune tree) ⟶ (1 (2 (3)) (4 (5)) (7 (8) (9 (10))))
```

Примерни решения

Haskell

```
leaf = null . subtrees

twig = all leaf . subtrees

stick (T _ [t]) = stick t
stick (T _ ts)  = null ts

trim (T x ts) = T x [ trim t | t <- ts, not $ twig t ]

prune t@(T x []) = t
prune t@(T x ts) = T x (if stick t
                        then let [T y _] = ts in [T y []]
                        else map prune ts)
```

Scheme

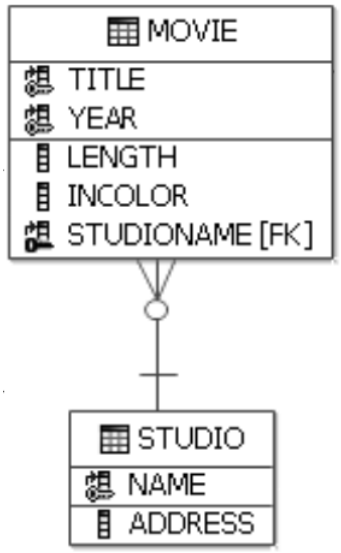
```
(define (leaf t)
  (null? (cdr t)))

(define (twig t)
  (null? (filter (lambda (x) (not (leaf x))) (cdr t))))

(define (trim t)
  (cons (car t) (map trim (filter (lambda (x) (not (twig x))) (cdr t)))))

(define (stick t)
  (or (leaf t) (and (null? (cddr t)) (stick (cadr t)))))

(define (prune t)
  (if (leaf t) t
      (cons (car t) (if (stick t)
                        (list (list (caadr t)))
                        (map prune (cdr t))))))
```

Критерии за оценяване

Ако е писано и по двата езика, взима се по-високият резултат. Сумата от точките се закръгля нагоре до цяло число.

- **0,5 т.** за коректна реализация на `leaf`;
  - **0 т.** за реализация на Haskell, която използва непълно съпоставяне с образци (pattern matching) само с `T _ []`, без да разглежда случая за непразен списък от директни поддървета;
- **1,5 т.** за коректна реализация на `twig`;
  - **0,5 т.** ако се прави проверка само за първото едно или две директни поддървета;
- **1,5 т.** за коректна реализация на `stick`;
- **3,0 т.** за коректна реализация на `trim`, от които:
  - **1,0 т.** за коректно извикване на `filter`;
  - **1,0 т.** за коректно извикване на `map`;
  - **1,0 т.** за коректна последователност на `map` след `filter`;
- по ред на празните слотове на `prune`:
  1. **0,5 т.** за `t` или `T x []` за Haskell;
  2. **1,5 т.** за коректно конструиране на листо със стойността на единственото дете на `t`;
  3. **1,5 т.** за коректно рекурсивно извикване на `prune` за всички директни поддървета на `t`.

Задача 5. Дадена е базата от данни `Movies`, в която се съхранява информация за филми и филмови студиа, които ги произвеждат.



Таблицата `Studio` съдържа информация за филмови студиа:

- `name` — име, първичен ключ
- `address` — адрес;

Таблицата `Movie` съдържа информация за филми. Атрибутите `title` и `year` заедно формират първичния ключ.

- `title` — заглавие
- `year` — година, в която е заснет филмът
- `length` — дължина в минути
- `incolor` — `'Y'` за цветен филм и `'N'` за черно-бял
- `studioname` — име на студио, външен ключ към `Studio.name`;

1) Да се напише заявка, която извежда заглавията и дължините в минути на всички цветни филми без най-дългия цветен и без тези с неизвестна дължина. Ако има няколко филма с максимална дължина, нито един от тях не трябва да бъде изведен.

2) Да се посочи коя от следните заявки извежда имената на тези студиа, които нямат филми или са снимали филми само в една единствена година:

А)
```
SELECT name
FROM Studio
WHERE name NOT IN (SELECT studioname
                   FROM Movie)
      OR COUNT(DISTINCT year) = 1;
```

Б)
```
SELECT studioname
FROM (SELECT studioname, year
      FROM Movie
      GROUP BY studioname, year) Years
GROUP BY studioname
HAVING COUNT(*) <= 1;
```

В)
```
SELECT name
FROM Studio
LEFT JOIN Movie ON name = studioname
GROUP BY name
HAVING COUNT(DISTINCT year) <= 1;
```

Г)
```
SELECT name
FROM Studio
WHERE NOT EXISTS (SELECT * FROM Movie)
UNION
SELECT studioname
FROM Movie
GROUP BY studioname
HAVING COUNT(DISTINCT year) = 1;
```

Примерно решение на подзадача 1:

```
SELECT title, length
FROM Movie
WHERE incolor = 'Y'
      AND length < (SELECT MAX(length)
                    FROM Movie
                    WHERE incolor = 'Y');
```

Критерии за оценяване:

1) Общо 5 т., от които
   - 1 т. за коректни `SELECT` и `FROM` клаузи на външната заявка, както и `WHERE` клауза на външната заявка без проверка на дължина;
   - 3 т. за коректна подзаявка за намиране на най-дългите цветни филми;
     - ако подзаявката намира най-дългите филми изобщо, се дава само 1 т. за този критерий;
     - ако подзаявката намира точно един най-дълъг филм, се дават само 2 т. за този критерий;
   - 1 т. за правилна релация с резултата от подзаявката (в конкретния пример това са `<`, `<>`, `!=`, `NOT IN`).

2) Единственият верен отговор е B). Посочването на този отговор носи 5 т., а посочването на грешен отговор или комбинация от отговори носи 0 т.

Задача 6. Години наред се провежда шахматно състезание, в което участват 256 шахматисти, които играят в турнир чрез директни елиминирания и в 8 етапа се определя победителят.

Тази година обаче, по някаква причина участниците са 257 и организаторите променят правилата. Състезанието вече не е турнир с директни елиминирания, защото броят на участниците не е точна степен на двойката. Новите правила изискват всеки шахматист да играе с точно 21 други участници точно по веднъж, и от резултатите от тези изиграни партии да се оформи крайното класиране.

Възможно ли е да се организира състезанието по новите правила и, ако да, по колко различни начина може да се случи това?

Отговорът носи точки само ако е придружен от коректна обосновка.

Примерно решение:

Отговорът е “не”. Ако моделираме състезанието с граф, чиито върхове са шахматистите, а ребрата са двойките шахматисти, които играят един срещу друг, този граф трябва да има 257 върха и да е 21-регулярен.

Такъв граф обаче не съществува, понеже във всеки граф, върховете от нечетна степен са четен брой. □

Схема за оценяване:

10 точки носи всяко решение, което показва невъзможността да се организира състезание с такива правила.

Най-лесно и естествено е задачата да се моделира с граф и да се използва наготово теоремата, гласяща, че броят на върховете от нечетна степен е четен. Не е необходимо да се доказва тази теорема. Ако обаче не се въвежда граф, а задачата се решава като комбинаторна задача чрез комбинаторните принципи, 10 точки се дават на решение, съдържащо подробно доказателство.

Частични точки се дават на решение, което съдържа доказателство с незначителни пропуски в прецизността.

Решение, което само казва, че няма начин да се проведе състезанието по новите правила, но няма аргументация, се оценява с 0 точки.

Всяко грешно като резултат решение се оценява с 0 точки.

Задача 7. Нека $\Sigma = \{0, 1\}$. За език $L \subseteq \Sigma^*$ казваме, че дума $w \in (\Sigma \cup \{\#\})^*$ е *L-половинчата*, ако всяка поддума на w от вида $v = \#u\#$ с $u \in \Sigma^*$ има свойството, че $u \cdot u \in L$.

Вярно ли е, че за всеки регулярен език $L \subseteq \Sigma^*$, езикът:

$$L_{\text{half}} = \{w \in (\Sigma \cup \{\#\})^* \mid w \text{ е } L\text{-половинчата}\}$$

е регулярен? Отговорът да се обоснове.

Примерно решение:

Да, вярно е!

Нека L е фиксиран регулярен език над Σ. Тъй като регулярните езици са затворени относно допълнение, достатъчно е да докажем, че $\overline{L_{\text{half}}} = (\Sigma')^* \setminus L_{\text{half}}$ е регулярен, където $\Sigma' = \{0, 1, \#\}$.

За дума $w \in \Sigma'^*$ е ясно, че $w \in \overline{L_{\text{half}}}$ точно когато $w \notin L_{\text{half}}$, тоест точно когато w съдържа поддума $v = \#u\#$, за която $u \in \Sigma^*$, но $u \cdot u \notin L$.

Тогава, ако означим с $L_2 = \{u \in \Sigma^* \mid u \cdot u \in L\}$, а $\overline{L_2} = \Sigma^* \setminus L_2$ получаваме, че:

$$\overline{L_{\text{half}}} = \Sigma'^* \cdot \{\#\} \cdot \overline{L_2} \cdot \{\#\} \cdot \Sigma'^*. \tag{1}$$

Тъй като регулярните езици са затворени относно конкатенация и итерация, а очевидно Σ' и $\{\#\}$ са регулярни, остава да покажем, че $\overline{L_2}$ е регулярен.

Но регулярните езици са затворени относно допълнение, така че е достатъчно да покажем, че L_2 е регулярен. За целта, нека $\mathcal{A} = \langle \Sigma, Q, s, \delta, F \rangle$ е тотален краен детерминиран автомат за езика L. От Теоремата на Клини такъв има, защото L е регулярен.

Тогава $u \in L_2$ точно когато $u \cdot u \in L$, тоест $\delta^*(s, u \cdot u) \in F$, точно когато $\delta^*(\delta^*(s, u), u) \in F$, откъдето получаваме, че

$$u \in L_2 \iff \exists q \in Q(\delta^*(s, u) = q \& \delta^*(q, u) \in F). \tag{2}$$

Нека $\overrightarrow{A}_q = \langle \Sigma, Q, q, \delta, F \rangle$, а $\overleftarrow{A}_q = \langle \Sigma, Q, s, \delta, \{q\} \rangle$. Тогава $\delta^*(s, u) = q$ е еквивалентно на $u \in L(\overleftarrow{A}_q)$, а $\delta^*(q, u) \in F$ – на $u \in L(\overrightarrow{A}_q)$.

Така получихме, че:

$$u \in L_2 \iff \exists q \in Q(u \in L(\overleftarrow{A}_q) \& u \in L(\overrightarrow{A}_q)) \iff u \in \bigcup_{q \in Q} (L(\overleftarrow{A}_q) \cap L(\overrightarrow{A}_q)). \tag{3}$$

Отново от Теоремата на Клини, езиците $L(\overleftarrow{A}_q)$ и $L(\overrightarrow{A}_q)$ са регулярни и тъй като регулярните езици са затворени относно допълнение и обединение, то езикът L_2 е регулярен. Оттук $\overline{L_2}$, а значи и $\overline{L_{\text{half}}}$ и L_{half} са регулярни.

Критерии за оценяване:

- 1 т. — за свеждане към $\overline{L_{\text{half}}}$;
- 2 т. — за (1), от които 1 т. — за описание с думи, съответно доказателство;
- 1 т. — за свеждане на $\overline{L_2}$ към L_2;
- 2 т. — за доказателство на (2);
- 1 т. — за въвеждане на $\overleftarrow{A}_q$ и $\overrightarrow{A}_q$;
- 2 т. — за доказателство на (3);
- 1 т. — за довършване.

Алтернативна схема за първата част: Езикът L_{half} може да се изрази и така:

$$\begin{aligned} L_{\text{half}} &= \Sigma^* \cup \Sigma^* \cdot \{\#\} \cdot (L_2 \cdot \{\#\})^* \cdot \Sigma^*, \text{ или} \\ L_{\text{half}} &= \Sigma^* \cup \Sigma^* \cdot (\{\#\} \cdot L_2)^* \cdot \{\#\} \cdot \Sigma^*, \text{ или} \\ L_{\text{half}} &= \Sigma^* \cdot (\{\varepsilon\} \cup \{\#\} \cdot (L_2 \cdot \{\#\})^*) \cdot \Sigma^* \text{ и т.н.} \end{aligned}$$

Изразявания като горните се оценяват с 4 точки, които съответстват на първите 4 точки от горната схема. Тези точки се разпределят така:

- 2 т. — за декларация на съответното регулярно изразяване;
- 2 т. — за доказателство, че съответното изразяване е коректно.

Забележка: Точки **не са дават** при грешно изразяване, например:

$$\begin{aligned} L_{\text{half}} &= \Sigma^* \cdot \{\#\} \cdot (L_2 \cdot \{\#\})^* \Sigma^*, \text{ или} \\ L_{\text{half}} &= \Sigma^* \cup \Sigma^* \cdot (\{\#\} \cdot L_2 \cdot \{\#\})^* \cdot \Sigma^*, \text{ или} \\ L_{\text{half}} &= \Sigma^* \cdot (\{\varepsilon\} \cup \{\#\} \cdot (L_2 \cdot \{\#\})^*) \text{ и т.н.} \end{aligned}$$

Решението обаче може да получи до 6 точки според оценяването за регулярността на езика L_2.

Забележка: Конструкции на крайни автомати, за които се твърди, че разпознават L_{half}, без едно от двете: (i) ясна семантика на състоянията или (ii) пълно доказателство за коректност, се оценяват с 0 точки. Самият едносричен отговор носи 0 точки.

Задача 8. Да се намери неопределеният интеграл

$$\int \left(x + \operatorname{tg}^2 x\right) \sin^2 x \, dx, \quad x \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right).$$

Примерно решение:

Започваме със свеждането

$$\int \left(x + \operatorname{tg}^2 x\right) \sin^2 x \, dx = \int x \sin^2 x \, dx + \int \frac{\sin^4 x}{\cos^2 x} \, dx. \qquad (*)$$

Намираме всеки от двата неопределени интеграла горе вдясно. За първия имаме

$$\begin{aligned}
\int x \sin^2 x \, dx &= \frac{1}{2} \int x(1 - \cos 2x) \, dx = \frac{1}{2} \int x \, dx - \frac{1}{4} \int x \cos 2x \, d(2x) \\
&= \frac{x^2}{4} + \text{const} - \frac{1}{4} \int x \, d(\sin 2x) \quad \text{(внасяме } \cos 2x \text{ под знака на диференциала)} \\
&= \frac{x^2}{4} + \text{const} - \frac{1}{4} \left(x \sin 2x - \int \sin 2x \, dx \right) \quad \text{(интегрираме по части)} \\
&= \frac{x^2}{4} - \frac{1}{4} x \sin 2x - \frac{1}{8} \cos 2x + \text{const}.
\end{aligned}$$

За втория интеграл вдясно на $(*)$ имаме

$$\begin{aligned}
\int \frac{\sin^4 x}{\cos^2 x} \, dx &= \int \frac{\left(1 - \cos^2 x\right)^2}{\cos^2 x} \, dx \\
&= \int \frac{dx}{\cos^2 x} - 2 \int dx + \int \cos^2 x \, dx \\
&= \operatorname{tg} x - 2x + \text{const} + \frac{1}{2} \int (1 + \cos 2x) \, dx \\
&= \operatorname{tg} x - 2x + \text{const} + \frac{1}{2} \int dx + \frac{1}{4} \int \cos 2x \, d(2x) \\
&= \operatorname{tg} x - 2x + \frac{x}{2} + \frac{1}{4} \sin 2x + \text{const} \\
&= \operatorname{tg} x - \frac{3x}{2} + \frac{1}{4} \sin 2x + \text{const}.
\end{aligned}$$

Окончателно получаваме

$$\int \left(x + \operatorname{tg}^2 x\right) \sin^2 x \, dx = \frac{x^2}{4} - \frac{3x}{2} - \frac{1}{4} x \sin 2x + \frac{1}{4} \sin 2x - \frac{1}{8} \cos 2x + \operatorname{tg} x + \text{const}.$$

Критерии за оценяване: Общо 10 т., от които:

- за намиране на $\int x \sin^2 x \, dx$: 4 т., в това число:
  - за понижаване на степента на sin: 1 т.,
  - за намиране на $\int x \, dx$: 1 т.,
  - интегриране по части: 2 т.;
- за намиране на $\int \frac{\sin^4 x}{\cos^2 x} \, dx$: 4 т., в това число:
  - подходящо разлагане на подинтегралната функция: 1 т.,
  - за намиране на $\int \frac{dx}{\cos^2 x}$: 1 т.,
  - за намиране на $\int dx$: 1 т.,
  - за намиране на $\int \cos^2 x \, dx$: 1 т.;
- окончателен отговор: 2 т. (при отсъствие на интеграционната константа **не** се присъждат).

Чернова