Р Е Ш Е Н И Я
на някои задачи от държавни изпити, проведени в СУ, ФМИ

Задача 1. (10 т.) Следната задача да се реши на един от езиците за програмиране C++ или Java. Дадена е `float` матрица `img` с размери `M` ≤ `10` реда и `N` ≤ `10` стълба. Напишете функция `subsample`, която получава като аргументи `M`, `N` и `img` и извежда на екрана матрица `s` с размери `(M+1)/2` и `(N+1)/2` (при целочислено деление), всеки елемент `s[i][j]` на която е равен на средноаритметичното от всички елементи `img[y][x]`, такива,че

`i*2` ≤ `y` ≤ `i*2+1` и `j*2` ≤ `x` ≤ `j*2+1`.

Например, при матрица `img`, представена таблично по следния начин:

1.0	2.0	3.0
4.5	6.5	7.5

функцията да извежда на екрана: `3.5  5.25`

Решение: Ще използваме програмния език C++.

```
#include <iostream.h>
#include <conio.h>

typedef float Image[10][10];

void subsample(int M, int N, Image img) {
  int i,j,y,x;
  Image s;
  for (i = 0; i < (M+1)/2; i++) {
    for (j = 0; j < (N+1)/2; j++) {
      float sum = 0.0;
      int cnt = 0;
      for (y = i * 2; (y <= i * 2 + 1) && (y < M); y++) {
        for (x = j * 2; (x <= j * 2 + 1) && (x < N); x++) {
          sum += img[y][x];
          cnt++;
        }
      }
      s[i][j] = sum/cnt;
    }
  }
  for (i = 0; i < (M+1)/2; i++) {
    for (j = 0; j < (N+1)/2; j++) {
      cout << s[i][j] << "\t";
    }
    cout << "\r\n";
  }
}

void main() {
  clrscr();
  Image img;
  img[0][0] = 1.0; img[0][1] = 2.0; img[0][2] = 3.0;
  img[1][0] = 4.5; img[1][1] = 6.5; img[1][2] = 7.5;
  subsample(2, 3, img);
}
```

Задача 2. (10 т.) Следната задача да се реши на един от езиците за програмиране C++ или Java. Да се обозначи явно на кой от двата езика е решавана задачата. При решението на задачата да не се използват библиотеки за работа със структури от данни.

а) Да се дефинира подходяща *индуктивна (рекурсивна)* структура от данни, позволяваща представянето в паметта на програмата на възел на дърво от цели числа (`int`), за което всеки връх може да има произволен брой наследници (0, 1 или повече).

б) Да се дефинира *рекурсивна* функция (или статичен метод)

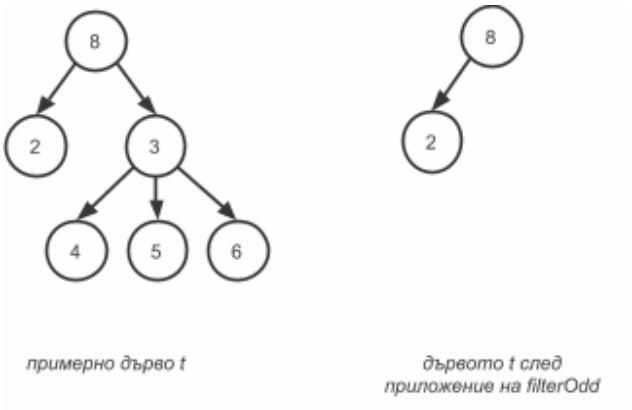
```
[булев тип] member ([подходящ тип]root, int x)
```

чиято стойност е истина точно тогава, когато в дървото с корен, представен от параметъра `root`, съществува възел със стойност `x`.

в) Да се дефинира рекурсивна функция (или статичен метод)

```
void filterOdd ([подходящ тип] root)
```

Функцията да премахва (чрез мутация) всяко поддърво `t'` на дървото с корен, представен от параметъра `root`, за което е изпълнено, че коренът на `t'` е със стойност нечетно число. На следната фигура е показано примерно дърво преди и след изпъпнението на операцията `filterOdd`.



Решение: Ще използваме програмния език C++.

```
#include <iostream.h>
#include <conio.h>
#include <alloc.h>

typedef int bool;

#define true 1
#define false 0

struct Node {
  int value;
  Node* nextSibling;
  Node* firstChild;
};

bool member(Node* root, int x) {
  if (root == NULL) return false;
  if (root->value == x) return true;
  root = root->firstChild;
  while (root != NULL) {
    if (member(root, x)) return true;
    root = root->nextSibling;
  }
  return false;
}

void freeTree(Node* root) {
  if (root == NULL) return;
  Node* current = root;
  root = root->firstChild;
  free(current);
  while (root != NULL) {
    current = root;
    root = root->nextSibling;
    free(current);
  }
}

void filterOdd(Node* root) {
  if (root == NULL) return;
  Node** prev = &(root->firstChild);
  root = root->firstChild;
  while (root != NULL) {
    Node* next = root->nextSibling;
    if ((root->value & 1) == 1) {
      *prev = next;
      freeTree(root);
    }
    else prev = &(root->nextSibling);
    root = next;
  }
}
```

```
void printTree(Node* root) {
  if (root == NULL)
    cout << "empty";
  else {
    cout << root->value;
    root = root->firstChild;
    bool hasChildren = (root != NULL);
    if (hasChildren) cout << " ( ";
    while (root != NULL) {
      printTree(root);
      cout << " ";
      root = root->nextSibling;
    }
    if (hasChildren) cout << ")";
  }
}

Node* newTree(int value, Node* firstChild, Node* nextSibling) {
  Node* root = (Node*) malloc(sizeof(Node));
  root->value = value;
  root->firstChild = firstChild;
  root->nextSibling = nextSibling;
  return root;
}

Node* createExampleTree() {
  return newTree(
    8,
    newTree(
      2,
      NULL,
      newTree(
        3,
        newTree(
          4,
          NULL,
          newTree(
            5,
            NULL,
            newTree(
              6,
              NULL,
              NULL
            )
          )
        ),
        NULL
      )
    ),
    NULL
  );
}
```

```
void testMembership(Node* root, int x) {
  if (member(root, x))
    cout << x << " is a member of the tree.\r\n";
  else
    cout << x << " is not a member of the tree.\r\n";
}

void main() {
  clrscr();
  Node* root = createExampleTree();
  cout << "Initial tree: ";
  printTree(root);
  cout << ".\r\n";
  testMembership(root, 3);
  testMembership(root, 6);
  testMembership(root, 7);
  testMembership(root, 2);
  cout << "\r\n";
  filterOdd(root);
  cout << "Filtered tree: ";
  printTree(root);
  cout << ".\r\n";
  testMembership(root, 3);
  testMembership(root, 6);
  testMembership(root, 7);
  testMembership(root, 2);
}
```

Задача 3. (10 т.) Да се построи минимален детерминиран краен автомат, еквивалентен на автомата

$A = \langle \{q0, q1, q2, q3, q4, q5, q6\}, \{0, 1\}, q0, \delta, \{q6\} \rangle$

със следната функция на преходите:

δ:

q	0	1
q_0	Ø	$\{q_0, q_3, q_6\}$
q_1	$\{q_2, q_6\}$	$\{q_5\}$
q_2	$\{q_2, q_6\}$	$\{q_1\}$
q_3	$\{q_3\}$	$\{q_3, q_4\ q_6\}$
q_4	Ø	Ø
q_5	Ø	$\{q_1\}$
q_6	Ø	Ø

Решение: Състоянието q_4 е излишно: от него не може да се стигне до финално състояние. Състоянията q_1, q_2 и q_5 са недостижими (от q_0). Множеството $\{q_0, q_3, q_6\}$ се разглежда като ново състояние q_{36}, което прави излишно q_6. Окончателно, получаваме детерминирания автомат $A_2 = \langle \{q_0, q_3, q_{36}\}, \{0, 1\}, q_0, \delta_2, \{q_{36}\} \rangle$.

Състоянията са минимум три според теоремата на Майхил-Нероуд, защото има поне три нееквивалентни думи: 0, 10 и 1. Следователно автоматът A_2 е минимален.

δ_2:

q	0	1
q_0	Ø	$\{q_{36}\}$
q_3	$\{q_3\}$	$\{q_{36}\}$
q_{36}	$\{q_3\}$	$\{q_{36}\}$

Задача 4.(10 точки). Даден е крайният автомат

$A = <\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}, \{a, b\}, q_0, \delta, \{q_6\}>$

със следната функция на преходите:

δ:

q	a	b
q_0	$\{q_1, q_3, q_6\}$	$\{q_4, q_6\}$
q_1	$\emptyset$	$\{q_2\}$
q_2	$\{q_5, q_6\}$	$\{q_2\}$
q_3	$\{q_5, q_6\}$	$\emptyset$
q_4	$\emptyset$	$\{q_4, q_6\}$
q_5	$\emptyset$	$\{q_5, q_6\}$
q_6	$\emptyset$	$\emptyset$

Да се построи минимален детерминиран краен автомат, еквивалентен на дадения.

Решение: Състоянията q_4 и q_5 функционират по един и същи начин, затова можем да се освободим от едното от тях; например заменяме q_4 с q_5 навсякъде (и изтриваме реда q_4). Множеството $\{ q_5 , q_6 \}$ заменяме с едно ново състояние q_{56}; то е финално, защото q_6 е такова. След добавянето на q_{56} състоянието q_5 става недостижимо. Множеството $\{q_1, q_3, q_6\}$ заменяме с ново финално състояние q_{136}, което прави недостижими състоянията q_1, q_3 и q_6. Окончателно, получаваме детерминирания автомат

$A_2 = <\{q_0, q_2, q_{56}, q_{136}\}, \{a, b\}, q_0, \delta_2 , \{ q_{56}, q_{136} \}>$.

Състоянията са минимум четири по теоремата на Майхил-Нероуд, защото има поне четири нееквивалентни думи: a, b, ab и ba. Следователно автоматът A_2 е минимален.

δ_2:

q	a	b
q_0	$\{ q_{136} \}$	$\{ q_{56} \}$
q_{136}	$\{ q_{56} \}$	$\{ q_2 \}$
q_2	$\{ q_{56} \}$	$\{ q_2 \}$
q_{56}	$\emptyset$	$\{ q_{56} \}$

Задача 5. (10 т.) Даден е неориентиран граф $G = (V, E)$ без примки. За всеки $u \in V$ съществуват точно три ребра от E, такива че u е връх в тях. Известно е, че G няма цикли с дължина 3.

а) (5 точки) Докажете, че G има поне 6 върха.

б) (5 точки) Има ли граф с 6 върха, изпълняващ условието на задачата? Ако няма такъв граф с 6, докажете това. Ако има такъв граф с 6 върха, опишете или нарисувайте този граф.

Решение:

а) Нека u_1 е един връх на G. По условие от излизат точно три ребра, които не са примки, т.е. водят към върхове u_2 , u_3 и u_4 , различни от u_1. Понеже G е граф, а не мултиграф, върховете u_2 , u_3 и u_4 са два по два различни един от друг. Доказахме, че G има поне четири върха.

Понеже в G няма цикли с дължина 3, то между никои два от върховете u_2 , u_3 и u_4 няма ребро.Да изберем един от тях, например u_4. От него излизат точно три ребра, а досега има само едно такова ребро ($u_4 – u_1$), то другите две ребра ($u_4 – u_5$ и $u_4 – u_6$) свързват u_4 с други два върха u_5 и u_6 , неразгледани досега. Следователно графът G има поне шест върха.

Задача 6. (10 точки). Нека $G=(V, E)$ е дърво. Говорейки за *път* в G имаме предвид прост път: такъв без повтаряне на върхове. Ако p е път в G, то $|p|$ означава дължината на p. Нека X е множеството от пътищата в G. Нека $Y \subseteq X$ е подмножеството на X, което се дефинира по следния начин.

За всеки път $p_1 \in Y$ и за всеки път $p_2 \in X$: $|p_1| \geq |p_2|$.

Докажете или опровергайте следните твърдения:

(а – 5 точки) За всеки два пътя $p, q \in Y$ съществува връх $u \in V$, такъв че u е общ за p и q.

(б – 5 точки) Съществува връх $u \in V$, такъв че за всеки два пътя $p, q \in Y$, u е общ за p и q.

Решение: И двете твърдения са верни.

а) Най-напред, ясно е, че Y е множеството от всички най-дълги пътища в дървото G. Нека p и q са два най-дълги пътя, L = | p | = | q | е максималната дължина на път от G. Нека A и B са краищата на p, C и D са краищата на q. Да допуснем, че p и q нямат общ връх. Следователно C ∉ p и A ∉ q. Понеже G е свързан граф, то има път r от A до C. Нека X е първият връх в r, който не е от p (такъв има, например C). Следователно всички върхове от r преди X принадлежат на p. Всички върхове от r след X не принадлежат на p (иначе някои ребра на r и p ще образуват цикъл, което е невъзможно, тъй като G е дърво). Аналогично, нека Y е последният връх от r, който не е от q. Тогава всички върхове от r, които са след Y, принадлежат на q , а тези преди Y не принадлежат на q.

Нека Z е върхът от r непосредствено преди X, а W е върхът от r непосредствено след Y. Тогава Z ∈ p, W ∈ q , а участъкът от r между Z и W не съдържа върхове нито от p, нито от q. Символично p = AZB, q = CWD (не всички върхове са посочени). Следователно

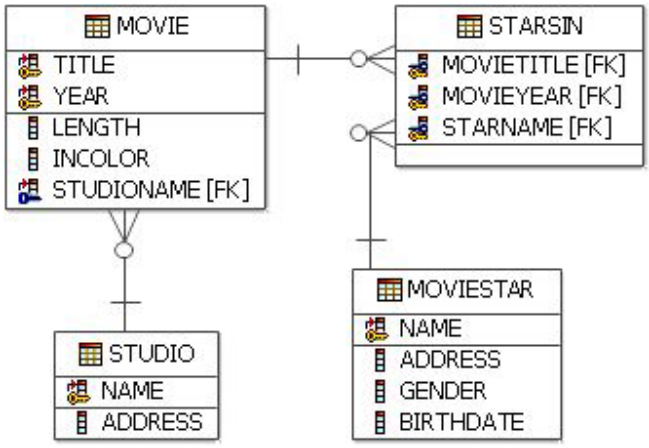
$$|AZ| + |ZB| = L \text{ и } |CW| + |WD| = L.$$

Във всеки от двата сбора едното събираемо е поне равно на половината от сбора.

Без ограничение нека $|AZ| \geq \frac{L}{2}$ и $|CW| \geq \frac{L}{2}$. Освен това, понеже p и q нямат общ връх, то Z и W са различни върхове, затова $|ZW| \geq 1$. Тогава AZWC е път с дължина $|AZ| + |ZW| + |WC| \geq \frac{L}{2} + 1 + \frac{L}{2} = L+1 > L$, което противоречи на максималността на L.

б) Ако в графа G има само един най-дълъг път, твърдението е тривиално вярно. Ако в G има точно два най-дълги пътя, твърдението се свежда до доказаното в подусловие “а”. Затова нека в G има поне три най-дълги пътя. Ще докажем, че твърдението е вярно чрез допускане на противното. Да допуснем, че някои три пътя p, q , r нямат общ връх. Обаче според доказаното в “а” всеки два от тях имат общ връх: A — за p и q , B — за q и r , C — за p и r. Тогава C ∉ q Следователно от A до B има поне два различни пътя: единият минава само по ребра от q , другият върви отначало (от A до C) по ребрата на p, а после (от C до B) върви по ребрата на r. Двата пътя са различни, защото единият съдържа C, а другият — не. (Това не пречи да имат общи върхове, различни от краищата, но така или иначе си остават два различни пътя.) Съществуването на два различни пътя поражда цикли, което противоречи на факта, че G е дърво.

Задача 7 (10 точки). Дадена е базата от данни Movies.



Таблицата ***Studio*** съдържа информация за филмови студиа:

name – име, първичен ключ;
address – адрес.

Таблицата ***Movie*** съдържа информация за филми. Колоните *title* и *year* заедно формират първичния ключ.

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

Таблицата ***MovieStar*** съдържа информация за филмови звезди:

name – име;
address – адрес;
gender – пол, **'M'** за мъж и **'F'** за жена;
birthdate – рождена дата.

Таблицата ***StarsIn*** съдържа информация за участието на филмовите звезди във филмите. Трите колони заедно формират първичния ключ. Колоните *movietitle* и *movieyear* образуват външен ключ към Movie.

movietitle – заглавие на филма;
movieyear – година на заснемане на филма;
starname – име на филмовата звезда, външен ключ.

1. Да се посочи заявката, която извежда имената на всички филмови звезди, чието име не завършва на “a” и са играли както в цветни, така и в черно-бели филми.

а)
```
SELECT name
FROM MovieStar, StarsIn, Movie
WHERE name = starName AND movieTitle = title AND movieYear = year
      AND name != '%a' AND inColor = 'y' AND inColor = 'n';
```

б)
```
SELECT MovieStar.name
FROM MovieStar
WHERE NOT (name LIKE '%a')
      AND name IN (SELECT starName
            FROM StarsIn
            JOIN Movie ON movieTitle = title AND movieYear = year
            WHERE inColor = 'y' OR inColor = 'n');
```

в)
```
SELECT DISTINCT starName
FROM StarsIn
INNER JOIN Movie ON movieTitle = title AND movieYear = year
WHERE starName NOT LIKE '%a' AND inColor = 'y'
      AND starName = (SELECT starName
            FROM StarsIn, Movie
            WHERE inColor = 'n');
```

г)
```
SELECT starName
FROM StarsIn
JOIN Movie ON movieTitle = title AND movieYear = year
WHERE starName NOT LIKE '%a' AND inColor = 'y'
INTERSECT
SELECT starName
FROM StarsIn
JOIN Movie ON movieTitle = title AND movieYear = year
WHERE inColor = 'n';
```

V

2. Посочете заявката, която извежда за всяка филмова звезда, играла в най-много 5 филма, следната информация:

- име;
- рождена година;
- брой студиа, с които е работила.

Ако за дадена звезда няма информация в какви филми е играла, за нея също да се извежда ред (с брой студиа, равен на 0).

а)
```
SELECT DISTINCT name, birthdate.year, COUNT(studioName)
FROM MovieStar, StarsIn, Movie
WHERE name = starname AND
      ((movieTitle = title AND movieYear = year) OR title IS
NULL)
GROUP BY name, birthdate.year
HAVING COUNT(title) <= 5;
```

б)
```
SELECT name, YEAR(birthdate), COUNT(DISTINCT studioName)
FROM MovieStar
LEFT OUTER JOIN StarsIn ON name = starname
LEFT JOIN Movie ON movieTitle = title AND movieYear = year
GROUP BY name
HAVING COUNT(title) <= 5;
```

V

в)
```
SELECT StarsIn.starname, YEAR(birthdate),
       COUNT(DISTINCT studioName)
FROM Movie
JOIN StarsIn ON movieTitle = title AND movieYear = year
RIGHT OUTER JOIN MovieStar ON MovieStar.name = StarsIn.starname
GROUP BY StarsIn.starname
HAVING COUNT(DISTINCT title) <= 5;
```

г)
```
SELECT name, year(birthdate), COUNT(SELECT DISTINCT studioName
      FROM Movie
      JOIN StarsIn ON title = movieTitle AND year = movieYear
      WHERE starname = name)
FROM MovieStar
HAVING COUNT(SELECT * FROM StarsIn WHERE starname = name) <= 5
ORDER BY name, year(birthdate);
```

Задача 8: Пресметнете определения интеграл $\int_0^{1/2} \arcsin x \ \mathrm{dx}$.

Решение: Интегрираме по части:

$$\int_0^{1/2} \arcsin x \ \mathrm{dx} = x \arcsin x \Big|_0^{1/2} - \int_0^{1/2} x \ \mathrm{d}(\arcsin x) = \frac{1}{2} \cdot \frac{\pi}{6} - \int_0^{1/2} \frac{x}{\sqrt{1-x^2}} \ \mathrm{dx} =$$

$$= \frac{\pi}{12} - \frac{1}{2} \int_0^{1/2} \frac{1}{\sqrt{1-x^2}} \ \mathrm{d}(x^2) = \frac{\pi}{12} + \frac{1}{2} \int_0^{1/2} \frac{1}{\sqrt{1-x^2}} \ \mathrm{d}(1-x^2) = \frac{\pi}{12} + \sqrt{1-x^2} \Big|_0^{1/2} =$$

$$= \frac{\pi}{12} + \sqrt{\frac{3}{4}} - 1 = \frac{\pi}{12} + \frac{\sqrt{3}}{2} - 1.$$

Задача 9: Намерете неопределения интеграл $\int \frac{x-2}{x(x^2+2)} \ \mathrm{dx}$.

Решение: Разлагаме дробната рационална функция в сбор от елементарни дроби; след това интегрираме всяко събираемо поотделно:

$$\int \frac{x-2}{x(x^2+2)} \ \mathrm{dx} = \int \frac{x}{x^2+2} \ \mathrm{dx} + \int \frac{1}{x^2+2} \ \mathrm{dx} - \int \frac{1}{x} \ \mathrm{dx}.$$

В първото събираемо внасяме x под знака на диференциала, а второто и третото събираемо са таблични интеграли. Окончателно, неопределеният интеграл е равен на

$$\frac{1}{2} \ln(x^2+2) + \frac{1}{\sqrt{2}} \operatorname{arctg} \frac{x}{\sqrt{2}} - \ln|x| + C.$$