

СОФИЙСКИ УНИВЕРСИТЕТ
“СВ. КЛИМЕНТ ОХРИДСКИ”



ФАКУЛТЕТ ПО МАТЕМАТИКА
И ИНФОРМАТИКА

ДЪРЖАВЕН ИЗПИТ

ЗА ПОЛУЧАВАНЕ НА ОКС “БАКАЛАВЪР ПО ИНФОРМАЦИОННИ СИСТЕМИ”

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 защити 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 защитават 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) Да се попълнят празните места в кода на функцията `removeWhitespace` така, че тя да премахва от непразния низ `str` всички `whitespace` символи.

```
bool isWhitespace(char c)
{
    return c == ' ' || c == '\t' ||
           c == '\r' || c == '\n';
}

_____ removeWhitespace(char* str)
{
    size_t read=0, write=0;

    while(str[read]){
        if (isWhitespace(_____))
            read_____ ;
        else
            str[write++] = _____ ;
    }

    str[_____] = '\0';

    return str;
}
```

2) Под всеки от фрагментите да се посочи какво ще изведе той на стандартния изход.

```
for (int i = 0; i < 10; ++i) {
    if (i % 2) continue;
    cout << i;
}
```

```
int i = 0x10;
cout << i;
```

```
int a=1,b=2,c=3;
cout << (a ? b : c);
```

```
char str[] = "abc";
char* p = str;
++p;
++*p;
cout << str;
```

3) Да се попълнят празните места в кода на функцията `pass` така, че функцията `bubbleSort` да сортира в нарастващ ред елементите на масива `arr` с размер `size`. Абстрахирайте се от това, че алгоритъмът, разписан по този начин, работи не-ефикасно.

```
void pass(int* arr,
          size_t size,
          bool& swappedAtLeastOnce)
{
    if (size _____ 1)
        return;

    if (arr[0] _____) {
        std::swap(arr[0], arr[1]);
        swappedAtLeastOnce = _____;
    }

    pass(_____,
        _____,
        _____);
}
```

```
void bubbleSort(int* arr, size_t size)
{
    bool swappedAtLeastOnce = false;
    pass(arr, size, swappedAtLeastOnce);
    if (swappedAtLeastOnce)
        bubbleSort(arr, size);
}
```

4) Да се посочи какво ще изведе на стандартния изход следният фрагмент.

```
int x = 2;
int arr[] = {10, 20, 30};
cout << "\nA: " << 5./x;
cout << "\nB: " << (x << 4);
cout << "\nC: " << arr[!x];
cout << "\nD: " << *(arr+x);
cout << "\nE: " << (2 + x++);
```

A: _____
B: _____
C: _____
D: _____
E: _____

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 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 (всяко коректно попълнено празно място носи 0,5 точки):

1. 2,5 точки
2. 2 точки
3. 3 точки
4. 2,5 точки

Примерно решение:

1)

```
bool isWhitespace(char c)
{
    return c == ' ' || c == '\r' ||
           c == '\t' || c == '\n';
}

char* removeWhitespace(char* str)
{
    size_t read=0, write=0;

    while(str[read]){
        if (isWhitespace(str[read]))
            read++;
        else
            str[write++] = str[read++];
    }

    str[write] = '\0';

    return str;
}
```

2)

- 02468
- 16
- 2
- acc

3)

```
void pass(int* arr,
          size_t size,
          bool& swappedAtLeastOnce)
{
    if (size <= 1)
        return;

    if (arr[0] > arr[1]) {
        std::swap(arr[0], arr[1]);
        swappedAtLeastOnce = true;
    }

    pass(arr + 1, size - 1, swappedAtLeastOnce);
}

void bubbleSort(int* arr, size_t size)
{
    bool swappedAtLeastOnce = false;
    pass(arr, size, swappedAtLeastOnce);
    if (swappedAtLeastOnce)
        bubbleSort(arr, size);
}
```

4)

- A: 2.5
- B: 32
- C: 10
- D: 30
- E: 4

Задача 2. Дадена е следната програмата на езика за програмиране C++, от която липсват части.

Класовете и шаблоните Inc, Square, Sum и Max описват едноместни функции от тип $f : T \rightarrow T$. Видът на конкретната функция се дефинира от метода value в съответния клас.

Класът Inc представя функцията $f : \text{double} \rightarrow \text{double}$, $f(x) = x + 1$.

Класът Square представя функцията $f : \text{double} \rightarrow \text{double}$, $f(x) = x^2$.

Шаблонът на клас Sum представя функцията $f : T \rightarrow T$, $f(x) = f_1(x) + \dots + f_k(x)$, където $f_1(x), \dots, f_k(x)$, $k \geq 0$ е списък от функции от тип $f_i : T \rightarrow T$. Дадена функция се добавя към списъка с функции на обект от клас Sum<T> чрез метода addFunction.

Шаблонът на клас Max представя функцията $f : T \rightarrow T$, $f(x) = \max\{f_1(x), \dots, f_k(x)\}$, където $f_1(x), \dots, f_k(x)$, $k > 0$ е списък от функции от тип $f_i : T \rightarrow T$. Дадена функция се добавя към списъка с функции на обект от клас Max<T> чрез метода addFunction.

Function е шаблон на абстрактен клас, който е базов за Inc, Square, Sum и Max.

Функцията main въвежда от стандартния вход числото x и извежда най-голямата измежду стойностите $x + 1$, x^2 и $x^2 + x + 1$.

Да се попълнят липсващите части в програмата. Да се приеме, че класовете Sum и Max не е нужно да правят копие на подадените им функции.

```
#include <vector>
#include <iostream>
template <typename T>
class Function
{ public:
    _____ T value(T) const _____
};
class Inc : _____
{ public:
    double value(double x) const { return x+1; }
};
class Square : _____
{ public:
    double value(double x) const { return x*x; }
};
```

```
_____

class Max : _____
{ private:
    std::vector<_____> functions;
public:
    void addFunction(_____ f)
    { functions.push_back(f); }
    T value(T x) const
    {
        if(_____ )
            throw "Function list is empty!";

        return _____;
    }
};
```

```
_____
};

_____

class Sum: _____
{ private:
    std::vector<_____> functions;
public:
    void addFunction(_____ f)
    { functions.push_back(f); }
    T value(T x) const
    {
        _____
        return _____;
    }
};

int main()
{
    Inc i; Square sq; Sum<double> s;
    //(x+1)+(x*x)
    s.addFunction(&i); s.addFunction(&sq);
    Max<double> m;
    //{x+1, x*x, (x+1)+(x*x)}
    m.addFunction(&i); m.addFunction(&sq);
    m.addFunction(&s);
    double x; std::cin >> x;
    std::cout << m.value(x) << std::endl;
}
```

Примерно решение

```
#include <vector>
#include <iostream>

template<typename T>
class Function
{ public:
    virtual T value(T) const = 0;
};
class Inc : public Function<double>
{ public:
    double value(double x) const { return x+1; }
};
class Square : public Function<double>
{ public:
    double value(double x) const { return x*x; }
};
template<typename T>
class Max : public Function<T>
{ private:
    std::vector<Function<T>*> functions;
public:
    void addFunction(Function<T> *f) { functions.push_back(f); }
    T value(T x) const
    {
        if(functions.size()<1)
            throw "Function list is empty!";
        T result = functions[0]->value(x);
        for(Function<T> *f : functions)
            result = std::max(result,f->value(x));
        return result;
    }
};
template<typename T>
class Sum : public Function<T>
{ private:
    std::vector<Function<T>*> functions;
public:
    void addFunction(Function<T> *f) { functions.push_back(f); }
    T value(T x) const
    {
        T result = 0;
        for(Function<T> *f : functions)
            result += f->value(x);
        return result;
    }
};
int main()
{
    Inc i; Square sq;
    Sum<double> s; Max<double> m;
    s.addFunction(&i); s.addFunction(&sq);
```

```

        m.addFunction(&i); m.addFunction(&sq); m.addFunction(&s);
        double x; std::cin >> x;
        std::cout << m.value(x) << std::endl;
    }

```

Критерии за оценяване

Задачата се оценява по долната схема, като сумата на точките се дели на 4 и се закръгля до цяло число.

#include <vector>	
#include <iostream>	
template <typename T>	
class Function	
{ public:	
_____ T value(T) _____	
[*]virtual ... = 0;	+4 т.
};	
class Inc : _____	
[*] public Function<double>	+2 т.
{ public:	
double value(double x) const { return x+1;	
}};	
class Square: _____	
[*] public Function<double>	+2 т.
{ public:	
double value(double x) const { return x*x;	
}};	

[*]template <typename T>	+2 т.
class Max : _____	
[*]public Function<T>	+2 т.
{ private:	
std::vector<_____> functions;	
[*] Function<T>*	+2 т.
public:	
void addFunction(_____f)	
[*] Function<T>*	+1 т.
{ functions.push_back(f); }	
T value(T x) const	
{	
if(_____)	
[*] functions.size()<1	+2 т.
throw "Function list is empty!";	
	return _____;
[*] всяко вярно решение	+8 т.
	}
	};

	[*]template <typename T>
	+2 т.
	class Sum : _____
	[*]public Function<T>
	+2 т.
	{ private:
	std::vector<_____> functions;
	[*] Function<T>*
	+2 т.
	public:
	void addFunction(_____f)
	[*] Function<T>*
	+1 т.
	{ functions.push_back(f); }
	T value(T x) const
	{
	return _____;
	[*] всяко вярно решение
	+ 8 т.
	}
	};
	int main()
	{
	Inc i; Square sq; Sum<double> s;
	//(x+1)+(x*x)
	s.addFunction(&i); s.addFunction(&sq);
	Max<double> m;
	//{x+1, x*x, (x+1)+(x*x)}
	m.addFunction(&i); m.addFunction(&sq);
	m.addFunction(&s);
	double x; std::cin >> x;
	std::cout << m.value(x) << std::endl;
	}

Задача 3. Нека векторите $\mathbf{a}_1 = (-1, 8, 8, 3 - \lambda, -1)$, $\mathbf{a}_2 = (1, -2, -3, 0, 0)$, $\mathbf{a}_3 = (0, -1, -2, 1, 0)$, $\mathbf{a}_4 = (0, -5, -3, 0, 1)$ и $\mathbf{a}_5 = (1, -7, -6, \lambda - 4, 1)$ принадлежат на $\mathbb{R}^5$.

- а) Да се намери рангът на системата вектори $\{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4, \mathbf{a}_5\} \subset \mathbb{R}^5$, в зависимост от стойностите на реалния параметър λ .
- б) Да се намери хомогенна система линейни уравнения, такава че пространството W от решенията ѝ да съвпада с линейната обвивка на векторите $\ell(\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4, \mathbf{a}_5)$, ако $\lambda = 4$.
- в) Да се намери за кои стойности на реалните параметри α и β векторът $\mathbf{a} = (11, \alpha, -1, -13, \beta)$ принадлежи на подпространството $W \leq \mathbb{R}^5$ от условие б).

Примерно решение

а) Рангът на системата вектори $\{a_1, a_2, a_3, a_4, a_5\} \subset \mathbb{R}^5$ е равен на ранга на матрицата A , образувана от вектор-редовете $\{a_1, a_2, a_3, a_4, a_5\}$. Извършваме елементарни Гаусови преобразувания върху матрицата A , които я преобразуват последователно до необходимия еквивалентен вид, без да променят търсения ранг.

$$\begin{aligned} A = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{pmatrix} &= \begin{pmatrix} -1 & 8 & 8 & 3-\lambda & -1 \\ 1 & -2 & -3 & 0 & 0 \\ 0 & -1 & -2 & 1 & 0 \\ 0 & -5 & -3 & 0 & 1 \\ 1 & -7 & -6 & \lambda-4 & 1 \end{pmatrix} \sim \begin{pmatrix} 1 & -2 & -3 & 0 & 0 \\ 0 & -1 & -2 & 1 & 0 \\ 0 & -5 & -3 & 0 & 1 \\ 1 & -7 & -6 & \lambda-4 & 1 \\ -1 & 8 & 8 & 3-\lambda & -1 \end{pmatrix} \\ &\sim \begin{pmatrix} 1 & -2 & -3 & 0 & 0 \\ 0 & -1 & -2 & 1 & 0 \\ 0 & 0 & 7 & -5 & 1 \\ 0 & -5 & -3 & \lambda-4 & 1 \\ 0 & 6 & 5 & 3-\lambda & -1 \end{pmatrix} \sim \begin{pmatrix} 1 & -2 & -3 & 0 & 0 \\ 0 & -1 & -2 & 1 & 0 \\ 0 & 0 & 7 & -5 & 1 \\ 0 & 0 & 7 & \lambda-9 & 1 \\ 0 & 0 & -7 & 9-\lambda & -1 \end{pmatrix} \\ &\sim \begin{pmatrix} 1 & -2 & -3 & 0 & 0 \\ 0 & -1 & -2 & 1 & 0 \\ 0 & 0 & 7 & -5 & 1 \\ 0 & 0 & 0 & \lambda-4 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2' \\ a_3' \\ a_4' \\ 0 \end{pmatrix} \end{aligned}$$

В резултат получаваме следните два случая:

- 1) ако $\lambda \neq 4$ следва, че $r(A) = 4$.
- 2) ако $\lambda = 4$ следва, че $r(A) = 3$.
- б) При $\lambda = 4$ имаме, че $\dim W = 3$, тъй като

$$W = \ell(a_1, a_2, a_3, a_4, a_5) = \{w = \mu_1 a_1 + \mu_2 a_2' + \mu_3 a_3' \mid \mu_i \in \mathbb{R}, i = 1, 2, 3\}.$$

Образуваме хомогенната система линейни уравнения (ХСЛУ) с коефициенти координатите на векторите a_1, a_2' и a_3' :

$$U : \begin{cases} x_1 - 2x_2 - 3x_3 = 0 \\ x_2 + 2x_3 - x_4 = 0 \\ 7x_3 - 5x_4 + x_5 = 0 \end{cases} .$$

(1)

Общият вид на решението на системата (1) има вида:

$$(2q - p, q - 2p, q, -7p + 5q),$$

а една нейна фундаментална система от решения (ФСР) е

$u_1 = (-1, -2, 1, 0, -7)$ и $u_2 = (2, 1, 0, 1, 5)$, т.е.

$$U = \{u = \alpha_1 u_1 + \alpha_2 u_2 \mid \alpha_i \in \mathbb{R}, i = 1, 2\}$$

и в частност $\dim U = 2$.

Тогава търсената ХСЛУ, такава че пространството от решенията ѝ съвпада с подпространството $W \leq \mathbb{R}^5$ е:

$$W : \begin{cases} x_1 + 2x_2 - x_3 + 7x_5 = 0 \\ 2x_1 + x_2 + x_4 + 5x_5 = 0 \end{cases} .$$

(2)

в) От условие б) имаме, че ХСЛУ (2) задава подпространството W . Тогава векторът $\mathbf{a} = (11, \alpha, -1, -13, \beta) \in W \Leftrightarrow$

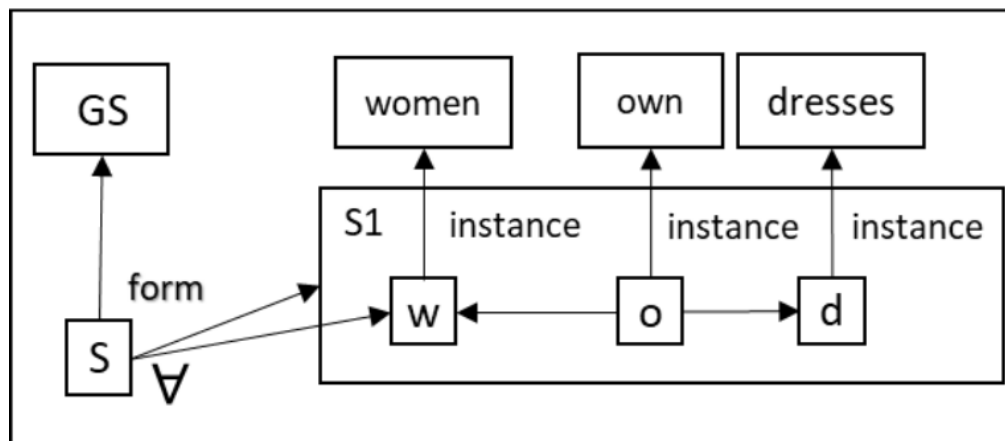
$$\begin{cases} 11 + 2\alpha + 1 + 7\beta = 0 \\ 22 + \alpha - 13 + 5\beta = 0 \end{cases} \Leftrightarrow \begin{cases} \alpha = 1 \\ \beta = -2 \end{cases}.$$

Следователно $\mathbf{a} = (11, 1, -1, -13, -2) \in W$.

Критерии за оценяване

- а) 3 т. при вярно и пълно решение;
- б) 4 т. при вярно и пълно решение;
- в) 3 т. при вярно и пълно решение.

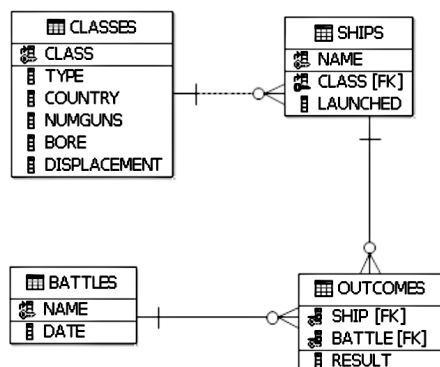
Задача 4. Да се представи чрез разделена семантична мрежа изречението „Every woman owns a dress.“

Примерно решение**Критерии за оценяване**

Дават се 10 т., в това число:

- за коректно представяне на основното твърдение – 3 т.
- за коректно конструиране на подмрежа и представяне на променливите в областта на действие на квантор за всеобщност – 4 т.
- за коректно свързване на подмрежата – 3 т.

Задача 5. 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 — броят на основните оръдия;
- bore — калибърът им (диаметърът на отвора на оръдието в инчове);

1) Да се напише заявка, която извежда без повторение имената на всички класове, от които няма нито един повреден в битка кораб. Ако даден клас няма никакви кораби или има, но те не са участвали в никакви битки, този клас също трябва да бъде изведен.

2) Да се посочи коя от следните заявки извежда имената на класовете и броя на потъналите кораби от съответния клас. Ако даден клас има кораби, но нито един от тях не е потънал или нито един от тях не е участвал в битка, срещу неговото име заявката да извежда числото 0. Ако даден клас няма никакви кораби, неговото име да НЕ се извежда.

A) `SELECT class, COUNT(ship)
FROM Ships
LEFT JOIN Outcomes
ON Ships.name = Outcomes.ship
AND result = 'sunk'
GROUP BY class;`

B) `SELECT class, COUNT(result = 'sunk')
FROM Ships
JOIN Outcomes
ON Ships.name = Outcomes.ship
GROUP BY class
HAVING COUNT(*) = 0;`

• 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 съдържа информация за битките:

- name — име на битката, първичен ключ;
- date — дата на провеждане.

Таблицата Outcomes съдържа информация за резултата от участието на даден кораб в дадена битка, като колоните ship и battle заедно формират първичния ключ:

- ship — име на кораба;
- battle — име на битката;
- result — резултат от битката: 'sunk' за потънал, 'damaged' за повреден и 'ok' за победил.

B) `SELECT DISTINCT class,
(SELECT COUNT(*)
FROM Outcomes
WHERE result = 'sunk')
FROM Ships;`

Г) `SELECT c.class, COUNT(DISTINCT result)
FROM Outcomes o
RIGHT JOIN Ships s ON o.ship = s.name
RIGHT JOIN Classes c
ON s.class = c.class
WHERE result = 'sunk'
GROUP BY c.class;`

Примерно решение на подзадача 1

```
SELECT class
FROM Classes
WHERE class NOT IN (SELECT class
                    FROM Ships
                    JOIN Outcomes ON name = ship
                    WHERE result = 'damaged');
```

Критерии за оценяване

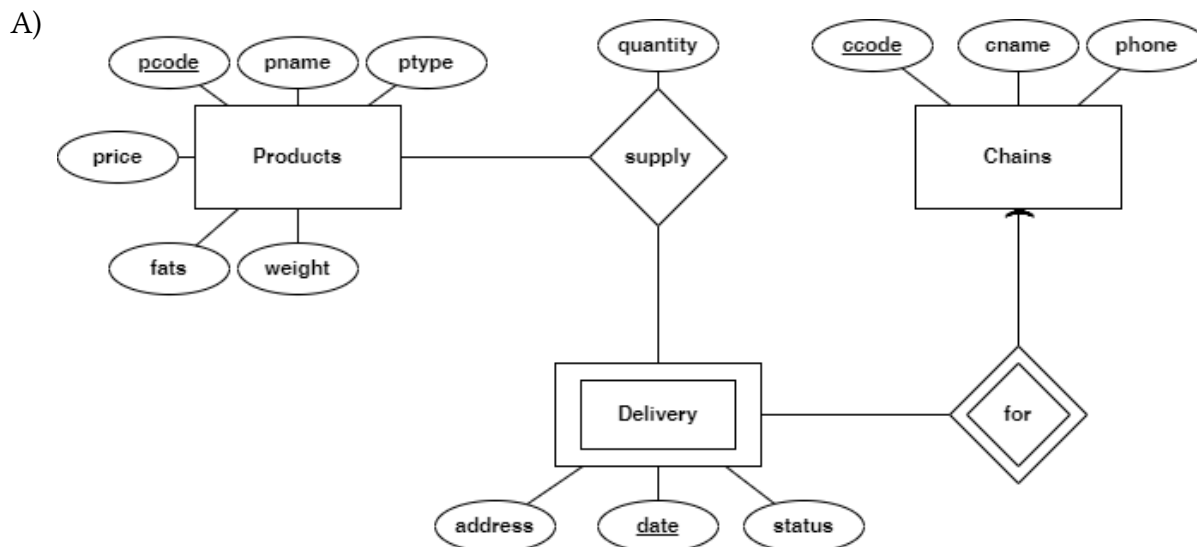
- 1) Общо 5 т., като:
 - решение, което коректно намира класовете, които имат поне един кораб, който е със статус, различен от 'damaged' се оценява с 2 т.
 - за всяка сгрешена клауза броят на точките се намалява с 2 до достигане на 0 т.
- 2) Единственият верен отговор е А). Посочването на този отговор носи 5 т., а посочването на грешен отговор или комбинация от отговори носи 0 т.

Задача 6. Информационна система съхранява информация за продуктите на фирмен магазин „Млечко“. Предлаганите от магазина продукти могат да бъдат: млека (прясно и кисело мляко), сирена (различни видове сирена и кашкавал) и други (масло, сладолед). Всеки продукт се характеризира с код на продукта (низ от точно 5 символа), който е уникален за всеки продукт и не може да бъде NULL, име на продукта (прясно мляко, кисело мляко, сирене, масло и т.н.) – низ до 30 символа, който не може да бъде NULL, масленост на продукта – ако е приложимо (реално положително число), тип на продукта (краве, овче, козе) - ако е приложимо (точно 1 символ), грамаж на продукта (цяло положително число), който не може да бъде NULL, и цена на продукта (реално положително число), която не може да бъде NULL. Фирменият магазин обслужва вериги от магазини, като прави доставки на продуктите си. Веригата от магазини се характеризира с код на веригата (низ от точно 3 символа), който е уникален за всяка верига и не може да бъде NULL, име на веригата (символен низ до 20 символа), което не може да бъде NULL, и телефонен номер (низ от точно 10 символа). Всяка доставка се определя еднозначно от конкретната верига и датата на доставка. Доставка до конкретна верига от магазини се извършва веднъж на ден. Като характеристика на доставката се пази и статус на доставката (в процес, приключена, отменена и т.н.) – низ от точно два символа и адрес (символен низ до 50 символа). На една верига от магазини могат да се доставят много продукти и един продукт може да бъде доставян на различни вериги от магазини. При доставката се определя и количеството на всеки продукт (цяло положително число), който ще бъде доставен в доставката към веригата, като количеството не може да бъде NULL.

Задание:

- А) Да се направи E/R модел на БД, която съхранява гореописаната информация. Да се начертае E/R диаграма на модела.
- Б) Да се преобразува E/R диаграмата към релационни схеми. Да се премахнат излишествата, където това е възможно.
- В) Да се напише DDL код, съответстващ на релационните схеми. Да се реализират всички описани ограничения.

Примерно решение



- Б) Products (pcode, pname, ptype, price, fats, weight)
 Chains (ccode, cname, phone)
 Delivery (ccode, date, status, address)
 Supply (pcode, ccode, date, quantity)

```

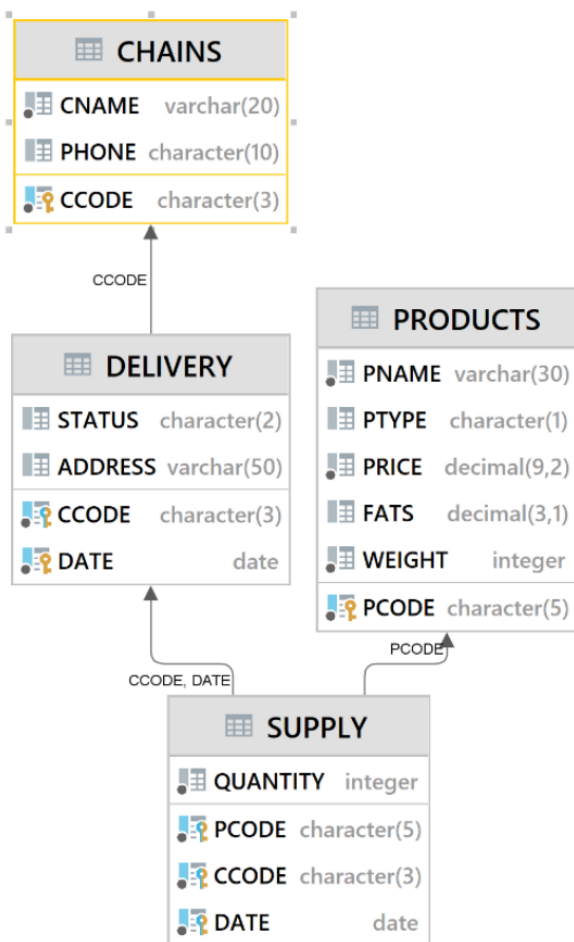
B) CREATE TABLE Products(
    pcode CHAR(5) NOT NULL PRIMARY KEY,
    pname VARCHAR(30) NOT NULL,
    ptype CHAR(1),
    price DECIMAL(9,2) NOT NULL CHECK(price > 0),
    fats DECIMAL(3,1) CHECK(fats > 0),
    weight INT NOT NULL CHECK(weight > 0)
);

CREATE TABLE Chains (
    ccode CHAR(3) NOT NULL PRIMARY KEY ,
    cname VARCHAR(20) NOT NULL,
    phone CHAR(10)
);

CREATE TABLE Delivery (
    ccode CHAR(3) NOT NULL REFERENCES Chains(CCODE),
    date DATE NOT NULL,
    status CHAR(2),
    address VARCHAR(50),
    PRIMARY KEY(ccode,date)
);

CREATE TABLE Supply (
    pcode CHAR(5) NOT NULL REFERENCES Products(pcode),
    ccode CHAR(3) NOT NULL,
    date DATE NOT NULL,
    quantity INT NOT NULL CHECK(quantity > 0),
    PRIMARY KEY (pcode, ccode, date),
  
```

```
FOREIGN KEY (ccode,date) REFERENCES Delivery(ccode,date)
);
```



Критерии за оценяване

- За грешки в Е/Р модела се взимат точки. Например, неотразена информация от условието на задачата, неправилно реализирана диаграма.
 - 1 точка се взима за неотбелязани първични ключове;
 - 1 точка се взима за неотбелязано слабо множество;
 - 1 точка се взима за неотбелязана поддържаща връзка;
 - 1 точки се взима за пропуснати атрибути или грешно използвана нотация на модела;
- За грешки при преобразуването от Е/Р към релационен модел се взимат точки.
 - 1 точка се взима за неоптимизирани връзки от тип „много към един“;
 - 1 точка се взима за неправилно преобразувано слабо множество;
 - 1 точка се взима за неправилно отбелязани първични ключове;
- За грешки в SQL синтаксиса се взимат точки.
 - 1 точка се взима за нереализирани ограничения – първични ключове и външни ключове;
 - 1 точка се взима за грешен синтаксис при създаването на таблиците;
 - 1 точка се взима за нереализирани ограничения – check ограничения и NOT NULL.

Задача 7. Разглеждаме информационната система на фирмен магазин „Млечко“, представена в Задача 6. Системата се ползва от служителите на „Млечко“, като им позволява да управляват зареждането на фирмения магазин с млечни продукти от производствената база, както и доставянето им от фирмения магазин към неговите клиенти – вериги от търговски обекти.

Служител в магазина получава продуктите в склада на магазина, като съпоставя опис на продуктите с доставеното. Продуктите може да идват от отменена поръчка, върната поръчка или получена доставка от производствената база. След това служителят намира подходящото място за всеки продукт в склада, поставя продукта на това място и актуализира информацията в системата.

Служител, който комуникира с клиентите, подготвя поръчки за доставка на желани от тях продукти. Поръчките се изпълняват автоматизирано ежедневно в определен час, като се проверява наличността на заявените продукти и се отразява в системата, че съответните количества от продукта се включват за изпълнение на поръчка. При липса на достатъчна наличност за всички поръчани продукти, наличните се включват за изпълнение, а за недостигащите се генерира поръчка за следващия ден.

След това поръчките се опаковат и се подготвят за изпращане. Служителят предава стоките заедно с опис и фактура на свой колега доставчик и отразява в системата, че продуктите са изпратени.

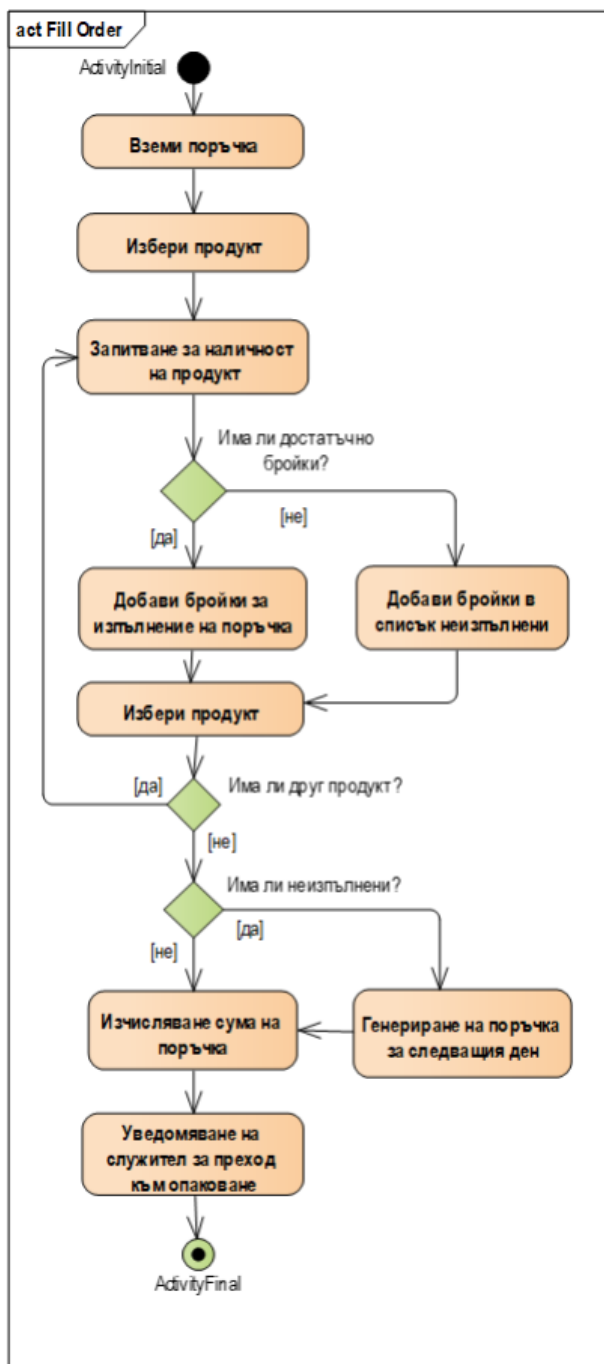
- а) Да се опише в пълен формат потребителски случай „Изпълнение на поръчка“. В описанието да се включат: актьори; предусловия; следусловия (резултати); основен сценарий; отношения с други потребителски случаи (ако е приложимо); алтернативен сценарий.
- б) Да се подготви диаграма на дейността за потребителския случай от подусловие а). Диаграмата да се подготви в термините на UML стандарта.

Примерно решение

а)

Име на ПС	Изпълнение на поръчка	
Актьори	Служител	
Предусловия	В резултат от изпълнението на ПС „Търсене на поръчка“ са изведени 1 или повече поръчки в състояние „За изпълнение“. Настъпил час за изпълнение на поръчки.	
Следусловия	Обект „Поръчка“ е в състояние „За опаковане“.	
Основен сценарий		
	1	Системата взима поръчка за изпълнение.
	2	Системата избира пореден продукт от обработваната поръчка.
	3	Системата проверява наличността на продукта.
	4	<i>Extension point (Има ли достатъчно бройки?) При не, преход към 4.1.</i>
	5	Системата добавя съответните бройки за изпълнение на поръчката.
	6	Системата избира пореден продукт от обработваната поръчка.
	7	<i>Extension point (Има ли други продукти?) При да, преход към 2.</i>
	8	<i>Extension point (Има ли неизпълнени?) При да, преход към 8.1.</i>
	9	Системата изчислява сумата на поръчката.
	10	Системата променя състоянието на обект „Поръчка“ на „За опаковане“.
	11	Системата уведомява служител за преход към опаковане.
Алтернативен сценарий 1		
	4.1	Системата добавя неизпълнения брой продукти в списък с неизпълнени.
	4.2	Изпълнението продължава в стъпка 5.
Алтернативен сценарий 2		
	8.1	Системата генерира поръчка за следващия ден с продукти с наличност по-малка от поръчаното количество.
	8.2	Изпълнението продължава в стъпка 9.

б)



Критерии за оценяване

- а) Максималният брой 5 точки се дава за решение, което изцяло покрива изискванията за съответствие, както следва:
- (0,30 т.) съответствие с описания сценарий;
 - (0,30 т.) съответствие с условието на задачата;
 - (4,40 т.) съответствие с изискванията за подготовка на модел на потребителските случаи (ПС) в частта пълно описание на ПС, включително:
 - (0,30 т.) коректно са определени актьори;

- (0,30 т.) коректно са определени предусловия и следусловия;
- (2,50 т.) коректно е определена функционалността на системата, която обхваща отделния потребителски случай;
- (0,30 т.) основен и алтернативен сценарии са описани в отделни стъпки и всеки от тях представлява напълно завършен сценарий;
- (1,00 т.) отношенията с други ПС са посочени в предусловие или на стъпките от сценария, където се извършва разклонение или извикване.

б) Максималният брой 5 точки се дава за решение, което изцяло покрива изискванията за съответствие, както следва:

- (3,00 т.) съответствие с описания сценарий;
- (0,50 т.) съответствие с условието на задачата;
- (1,00 т.) съответствие с UML нотацията;
- (0,50 т.) съответствие с описанието на ПС от подусловие а).

Сумата от точките се закръгля до цяло число.

Задача 8. В урна има шест топки, номерирани с числата 1, 2, ..., 6. От урната три пъти последователно се вади по една топка без връщане. Дефинираме събития:

$A = \{\text{първата топка има по-голям номер от втората}\};$

$B = \{\text{номерата на топките намаляват в реда на изваждане}\}.$

- а) Да се определят вероятностите на всяко едно от събитията.
 - б) Каква е вероятността да се изпълни B , ако знаем че се е изпълнило A ?
 - в) Нека опитът с изваждането на три топки от урна се повтаря многократно. Колко опита трябва да бъдат извършени, така че вероятността A да се сбъдне поне веднъж да е по-голяма от 0,999?
-

Примерно решение

- а) Вероятност на A : Събитието зависи само от първите две топки, така че третата топка няма значение. Всички начини за избиране на две топки с наредба без повторение са $V_6^2 = 6 \cdot 5 = 30$. Топките са различни, тогава от съображения за симетрия в половината случай първата топка ще е по-голяма, т.е. общо в 15 случая.

$$P(A) = \frac{1}{2}$$

Вероятност на B : Аналогично на A всички начини за избиране на трите топки са $V_6^3 = 6 \cdot 5 \cdot 4 = 120$.

Ще пресметнем броя на всички възможни намаляващи редици. Елементите, които ще участват в тази редица, ако не обръщаме внимание на подредбата, са три избрани измежду шест възможни без повторение, т.е. комбинации C_6^3 . След като знаем кои са елементите, можем да ги подредим в намаляваща редица по един единствен начин. Тогава броя на намаляващите редици е C_6^3 .

$$P(B) = \frac{C_6^3}{V_6^3} = \frac{1}{6}$$

- б) Винаги когато се изпълнява B , се изпълнява и A , тогава $B \subset A$ и следователно $A \cap B = B$. От формулата за условна вероятност получаваме:

$$P(B|A) = \frac{P(A \cap B)}{P(A)} = \frac{P(B)}{P(A)} = \frac{1}{3}$$

- в) С q ще означим вероятността събитието A да не се сбъдне при провеждането на един опит. Тогава, q^n е вероятността A да не се изпълни нито веднъж при провеждането на n опита. Противоположното събитие, т.е. A да се сбъдне поне веднъж е с вероятност $1 - q^n$. В задачата търсим:

$$1 - q^n > 0,999$$

$$q^n < 0.001$$

В нашия случай $q = 1 - P(A) = 1/2$ и от горното уравнение следва $2^n > 1000$. Известно е, че $2^{10} = 1024$, следователно отговорът е $n = 10$.

Критерии за оценяване

- а) 4 т. при вярно и пълно решение;
- б) 2 т. при вярно и пълно решение;
- в) 4 т. при вярно и пълно решение.

Чернова