СОФИЙСКИ УНИВЕРСИТЕТ
“СВ. КЛИМЕНТ ОХРИДСКИ”

ФАКУЛТЕТ ПО МАТЕМАТИКА
И ИНФОРМАТИКА

ДЪРЖАВЕН ИЗПИТ

ЗА ПОЛУЧАВАНЕ НА ОКС “БАКАЛАВЪР ПО СОФТУЕРНО ИНЖЕНЕРСТВО”

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) 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 неефикасно.

```
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 $f : T \to T$. Видът на конкретната функция се дефинира от метода `value` в съответния клас.

Класът `Inc` представя функцията f : `double` $\to$ `double`, $f(x) = x + 1$.

Класът `Square` представя функцията f : `double` $\to$ `double`, $f(x) = x^2$.

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

Шаблонът на клас `Max` представя функцията $f : T \to T$, $f(x) = max\{f_1(x), ..., f_k(x)\}$, където $f_1(x), ..., f_k(x)$, $k > 0$ е списък от функции от тип $f_i : T \to T$. 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 $\{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4, \mathbf{a}_5\} \subset \mathbb{R}^5$ е равен на ранга на матрицата A, образувана от вектор-редовете $\{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4, \mathbf{a}_5\}$. Извършваме елементарни Гаусови преобразувания върху матрицата A, които я преобразуват последователно до необходимия еквивалентен вид, без да променят търсения ранг.

$$A = \begin{pmatrix}\mathbf{a}_1\\ \mathbf{a}_2\\ \mathbf{a}_3\\ \mathbf{a}_4\\ \mathbf{a}_5\end{pmatrix} = \begin{pmatrix}-1 & 8 & 8 & 3-\lambda & -1\\ 1 & -2 & -3 & 0 & 0\\ 0 & -1 & -2 & 1 & 0\\ 0 & -5 & -3 & 0 & 1\\ 1 & -7 & -6 & \lambda-4 & 1\end{pmatrix} \sim \begin{pmatrix}1 & -2 & -3 & 0 & 0\\ 0 & -1 & -2 & 1 & 0\\ 0 & -5 & -3 & 0 & 1\\ 1 & -7 & -6 & \lambda-4 & 1\\ -1 & 8 & 8 & 3-\lambda & -1\end{pmatrix}$$
$$\sim \begin{pmatrix}1 & -2 & -3 & 0 & 0\\ 0 & -1 & -2 & 1 & 0\\ 0 & 0 & 7 & -5 & 1\\ 0 & -5 & -3 & \lambda-4 & 1\\ 0 & 6 & 5 & 3-\lambda & -1\end{pmatrix} \sim \begin{pmatrix}1 & -2 & -3 & 0 & 0\\ 0 & -1 & -2 & 1 & 0\\ 0 & 0 & 7 & -5 & 1\\ 0 & 0 & 7 & \lambda-9 & 1\\ 0 & 0 & -7 & 9-\lambda & -1\end{pmatrix}$$
$$\sim \begin{pmatrix}1 & -2 & -3 & 0 & 0\\ 0 & -1 & -2 & 1 & 0\\ 0 & 0 & 7 & -5 & 1\\ 0 & 0 & 0 & \lambda-4 & 0\\ 0 & 0 & 0 & 0 & 0\end{pmatrix} = \begin{pmatrix}\mathbf{a}_1\\ \mathbf{a}'_2\\ \mathbf{a}'_3\\ \mathbf{a}'_4\\ \mathbf{0}\end{pmatrix}$$

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

$$W = \ell(\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4, \mathbf{a}_5) = \{w = \mu_1\mathbf{a}_1 + \mu_2\mathbf{a}'_2 + \mu_3\mathbf{a}'_3 \mid \mu_i \in \mathbb{R}, i = 1, 2, 3\}.$$

Образуваме хомогенната система линейни уравнения (ХСЛУ) с коефициенти координатите на векторите a_1, a'_2 и a'_3:

$$U: \left|\begin{array}{rcrcrcrcl} x_1 & - & 2x_2 & - & 3x_3 & & & = & 0\\ & & x_2 & + & 2x_3 & - & x_4 & = & 0\\ & & & & 7x_3 & - & 5x_4 & + & x_5 = 0 \end{array}\right. . \tag{1}$$

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

$\mathbf{u}_1 = (-1, -2, 1, 0, -7)$ и $\mathbf{u}_2 = (2, 1, 0, 1, 5)$, т.е.

$$U = \{u = \alpha_1\mathbf{u}_1 + \alpha_2\mathbf{u}_2 |\ \alpha_i \in \mathbb{R}, i = 1, 2\}$$

и в частност $\dim U = 2$.

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

$$W: \left|\begin{array}{rcrcrcrcl} x_1 & + & 2x_2 & - & x_3 & & + & 7x_5 & = 0\\ 2x_1 & + & x_2 & & & + & x_4 & + & 5x_5 = 0 \end{array}\right. . \tag{2}$$

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

$$\left|\begin{array}{rcrcrcrcl} 11 & + & 2\alpha & + & 1 & & & + & 7\beta & = 0 \\ 22 & + & \alpha & & & - & 13 & + & 5\beta & = 0 \end{array}\right| \Leftrightarrow \left|\begin{array}{rcl} \alpha & = & 1 \\ \beta & = & -2 \end{array}\right| .$$

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. Спецификация на електронен магазин включва формуляр за завършване на поръчка и указване на начин на плащане и доставка. Формулярът включва попълване на 2 полета за избор на начин на плащане: „Наложен платеж“ или „Онлайн банкиране“, и тип доставка: „Адрес“ или „Офис на куриер“. В зависимост от категорията на клиента (притежател на Бронзова карта, Сребърна карта или Златна карта) и поръчаното количеството се начислява отстъпка (съответно 0%, 5% или 10%).

Да се дефинират тестови сценарии, покриващи избора на стойности от полетата за избор, чрез прилагане на техниката за тестване по двойки с ортогонален масив. Да се опишат стъпките за прилагане на техниката и да се представят тестовите сценарии в таблица.

Критерии за оценяване

- Избор на подходящ ортогонален масив: 2 т.
- Описани стъпки за прилагане на техниката за тестване: 4 т.
- Дефинирани тестови сценарии в таблица спрямо избрания ортогонален масив: 4 т.

Примерно решение

Стъпка 1: Независимите променливи са 4 (Factors).

Стъпка 2: Всяка променлива може да има най-много 3 стойности (Levels).

Стъпка 3: Избира се ортогонален масив $L_9(3^4)$.

Избира се следния ортогонален масив:

$L_9(3^4)$				
	Фактори			
	1	2	3	4
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1

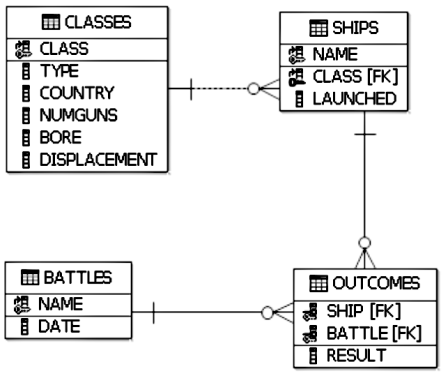
Стъпка 4: Свързване на променливите с факторите и стойностите с нивата на масива.

Стъпка 5: Фактор 2 и Фактор 4 имат три специфицирани нива в масива, но съответстващите им променливи има само 2 възможни стойности. Неасоциираните нива се запълват със стойностите на променливата, като се редуват отгоре надолу.

Стъпка 6: Генерират се 9 тестови сценария.

ТС	**Категория на клиента**	**Начин на плащане**	**Отстъпка**	**Тип доставка**
1	Бронзова карта	Наложен платеж	0%	Адрес
2	Бронзова карта	2 (Наложен платеж)	5%	Офис на куриер
3	Бронзова карта	Онлайн банкиране	10%	3 (Адрес)
4	Сребърна карта	Наложен платеж	5%	3 (Офис на куриер)
5	Сребърна карта	2 (Онлайн банкиране)	10%	Адрес
6	Сребърна карта	Онлайн банкиране	0%	Офис на куриер
7	Златна карта	Наложен платеж	10%	Офис на куриер
8	Златна карта	2 (Наложен платеж)	0%	3 (Адрес)
9	Златна карта	Онлайн банкиране	5%	Адрес

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);
- 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 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.

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.

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

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

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

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. Да се направи декомпозиция на модулите от архитектурата на софтуерна система за управление и оптимизиране на дейности в земеделско стопанство (напр. поливане, предсказване на проблеми и др.), според дадените по-долу изисквания. Да се обоснове защо така проектираната архитектура удовлетворява изискванията.

R1. Системата се състои от две части – сървърна и полева част.

- На сървърната част се съхраняват всички данни и потребителски профили.
- Полевата част на системата включва сензори, дронове и други устройства.

R2. Системата има два вида потребителски профили.

- Обикновени потребители, които може да имат една или повече ферми които да управляват чрез системата.
- Администратори, които одобряват регистрацията на потребителите и следят за изправността на системата.

R3. Сензорите осигуряват измервания (напр. атмосферна влажност и температура), които се изпращат към сървърната част на системата.

R4. Потребителите трябва да управляват отдалечено поливането на различни площи във фермата.

R5. Системата да дава възможност на потребителите да наблюдават в реално време данните от сензорите и да генерират от тях справки за различни периоди от времето.

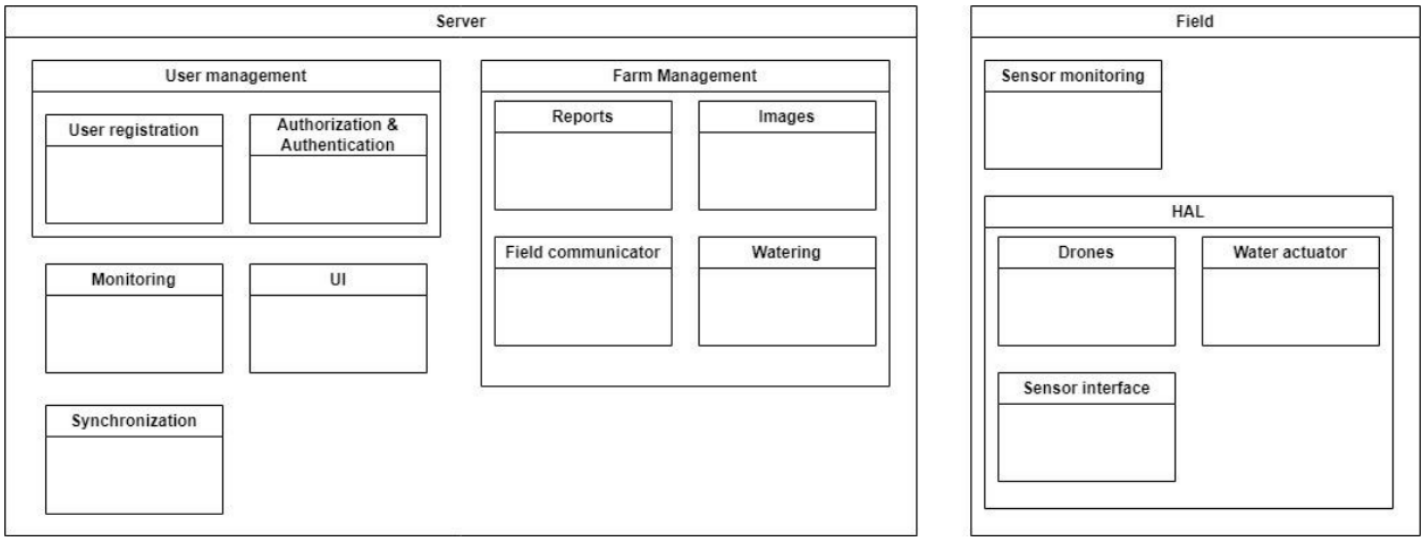
R6. На базата на предадените данни, системата да може да предлага оптимизирани планове за поливане на фермата.

R7. Системата да следи състоянието на сензорите и при отказ на някой от тях да се уведомява администратор.

R8. Чрез дроновете може да се снимат различни части на фермата и тези снимки да бъдат анализирани от външна система (напр. за по-ранна прогноза на болести по културите и т.н.)

R9. Трябва да има възможност за лесно добавяне на нови видове сензори (например за осветеност и т.н.).

R10. Допуска се профилактика веднъж годишно за 24 часа. През останалото време, системата трябва да е 99,999% налична.



Фигура 1: Декомпозиция на модулите

Примерно решение

Декомпозицията на модулите на системата е представена на фиг. 1. Според условието на задачата, системата е разделена на две подсистеми – полева част (Field) и сървърна част (Server).

Всички данни, с които работи системата, се съхраняват в базата данни, като връзката с нея се осъществява чрез модула DB Connector. Модулът User Manager осигурява функционалност за регистрация на потребителите (User Registration), вписване в системата, ролите им и правата за достъп (Authorization & Authentication). С това е изпълнено изискване R2.

В модула UI е реализирана логиката по потребителския интерфейс към системата.

Основната бизнес логика е реализирана в модула Farm Management. Модулът Reports отговаря за генерирането на различните видове справки, наблюдението на данните от сензорите в реално време и предложенията на планове за поливане на фермата. С това са удовлетворени изисквания R5 и R6. Чрез модула Field communicator се осъществява комуникацията с полевата част и също така се изпращат данни за отдалечено поливане на площите на фермата, с което е удовлетворено изискване R4.

Monitoring модулът в сървърната част има задача да актуализира потребителския интерфейс при наличие на сигнал от модула Sensor monitoring (в полевата част). Sensor monitoring от своя страна регистрира откази на сензорите на базата на тактика за следене на сигнали за активност от сензорите (heartbeat/keepalive) – ако някой от сензорите не изпрати данни след като е изтекъл определен интервал от време, се счита че има отказ. С това се удовлетворява изискване R7.

Модулът Hardware adaptation layer (HAL) служи като посредник за връзката със сензорите дроновете и изпълнителните механизми за поливане на площите във фермата (actuators). Подмодулът sensors осигурява връзката със сензорите и изпраща измерените от тях стойности към сървърната част, с което се удовлетворява изискване R3. Новите сензори се регистрират чрез уникален идентификатор, като за целта се прави нов запис в базата данни. По този начин се удовлетворява изискване R9.

Данните от направените изображения с дроновете се изпращат към модула Images в сървърната част, в който ако се налага се прави допълнителна обработка на изображенията, след което те се изпращат за анализ към външната подсистема. С това се удовлетворява изискване R8.

Изискване R10 може да се удовлетвори като се направи репликация на сървъра и базата данни, като модулът Synchronization има задача да осигури консистентност на данните между различните активни копия на системата.

Критерии за оценяване

Удовлетворено е изискването за свързване на сензорите	1 т.
Удовлетворено е изискването за добавяне на сензори	2 т.
Удовлетворени са изискванията за управление на поливането, справки и план за поливане	2 т.
Удовлетворено е изискването за включване на дронове в системата	1 т.
Удовлетворени са изискванията за съхранение и достъп до данните	2 т.
Удовлетворени са изискванията за наличност в системата	2 т.

Задача 7. Вашата софтуерна фирма е сключила договор с клиент (училище) за осъществяване на дигиталната му трансформация, касаеща управление на училището и учебния процес през дигитална платформа. Да се подготви план за работа, в който:

1. да се дефинират работни пакети и да се посочат задачите в тях, като всяка задача трябва да води до специфичен краен резултат, който може да се валидира и контролира от клиента;
2. за всеки работен пакет да се определи какви материални и нематериални ресурси и експертиза са необходими за този проект (директни разходи на фирмата).

Примерно решение

1. Работни пакети:
   - **РП1 Анализ на областта:** Проучване на теорията и практиката за управление на училището и учебния процес; Моделиране на работните процесите на училището; Събиране на информация за учебните предмети и спецификата им; Формулиране на изисквания (функционални и нефункционални) към платформата;
   - **РП2 Разработване на дигиталната платформа:** Проектиране на системата; Програмиране и тестване на системата; Разработване на база данни и интеграция с регистър на учители и ученици; Разработване на електронни материали (уроци и тестове) от учителите и попълване на базата данни
   - **РП3 Комуникация с клиента (учители и ръководители):** Семинар за валидиране на изискванията към платформата; Тестване на прототип на платформата от училището; Семинар за обучение на учители и ръководители от училището
   - **РП4 Управление на проекта:** Планиране и контрол; Финансово управление; Срещи и комуникация на ръководния екип; Отчетност и документация
2. Материални и нематериални ресурси и експертиза:
   - **РП1:** образователен експерт; бизнес анализатори; ИТ експерти, офис консумативи
   - **РП2:** програмисти (например: архитекти, дизайнери, тестери и др.), закупуване на сървър и техника за платформата, евентуално закупуване на специализиран образователен софтуер,
   - **РП3:** организатори, лектори, зали с мултимедия, кетъринг, материали за участниците, офис консумативи
   - **РП4:** мениджър, финансист, административен персонал, офис консумативи

Критерии за оценяване

1. Дефинирани работни пакети с посочени задачи в тях, като всяка задача води до специфичен краен резултат, който може да се валидира и контролира от клиента – макс. 6 точки.
2. За всеки работен пакет са определени необходимите за проекта материални и нематериални ресурси и експертиза – макс. 4 точки.

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 = {първата топка има по-голям номер от втората};

B = {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 $V_6^2 = 6.5 = 30$. 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 $V_6^3 = 6.5.4 = 120$.

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