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++.

Под всеки от дадените по-долу фрагменти да се посочи какво ще изведе той на стандартния изход. Упътване: *ASCII кодовете на символите `'A'` и `'a'` са съответно 65 и 97.*

1)

```
void print(char* p) {
  if (*(p++)) print(++p);
  std::cout << *(--p);
}
void main() {
   char str[] = "012345";
   print(str);
}
```

Отговор: _____

2)

```
const size_t size = 3;
int matrix[size][size] = {
  1, 2, 3,
  4, 5, 6,
  7, 8, 9
};
std::cout << matrix[1][2] << *matrix[2];
```

Отговор: _____

```
std::cout << **matrix << *(*(matrix+2)+1);
```

Отговор: _____

3)

```
char symb = 'A';
switch(symb) {
  case 0   : std::cout << "0";
  case 65  : std::cout << "1";
  case 'a' : std::cout << "2";
  default  : std::cout << "d";
}
```

Отговор: _____

4)

```
unsigned x = 8, y = 3;
while (x > 0) {
  if (x % y == 0) break;
  else x -= 1;
}
std::cout << x;
```

Отговор: _____

5)

```
unsigned a = 4, b = 8, c = 2;
std::cout << ((a > c) ? (b > a ? b : a) : c);
```

Отговор: _____

6) Да се попълнят празните места в кода на функцията `insert` така, че функцията `insertSort` да сортира в нарастващ ред елементите на масива `arr` с размер `size`.

```
void insert(int* arr, int index) {
 int key = arr[index];
 int j = _____;
 while (_____ && _____) {

    _____;
    j = j - 1;
  }
  arr[j + 1] = key;
}
void insertSort(int* arr, int size){
  for(int i = 1; i < size; i++) {
    insert(arr, i);
  }
}
```

7) Да се попълнят празните места във функцията така, че `rotateArray` да завърта с `k` стъпки надясно елементите на масива `arr` с размер `size`. Стандартната функция `std::reverse(start, end)` обръща реда на елементите на масив, зададен чрез указатели `start` към началото му и `end`, сочещ **след** последния му елемент.
Пример: *При nums = {1, -2, 13, 40, 5} и k = 2 резултатът е {40, 5, 1, -2, 13}.*

```
void rotateArray(int* nums,
  int k, int size) {
  if (size == 0 || k % size == 0)
    return;
  k %= size;
  std::reverse(nums, nums+size);
  std::reverse(_____, _____);
  std::reverse(_____, _____);
}
```

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 довършване на кода на функциите, 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. Изцяло правилно попълнен отговор носи 1 точка.
2. Всяко коректно попълнено празно място носи по 0.5 точки (общо 1 точка).
3. Изцяло правилно попълнен отговор носи 1 точка.
4. Изцяло правилно попълнен отговор носи 1 точка.
5. Изцяло правилно попълнен отговор носи 1 точка.
6. Всеки изцяло коректно попълнен ред носи по 1 точка (общо 3 точки).
7. Всяко изцяло коректно попълнено извикване на функцията носи по 1 точка (общо 2 точки).

Примерно решение:

1) Изцяло правилно попълнен отговор носи 1 точка.
Отговор: 531

2) Всяко коректно попълнено празно място носи по 0.5 точки (общо 1 точка).
Отговор: 67
Отговор: 18

3) Изцяло правилно попълнен отговор носи 1 точка.
Отговор: 12d

4) Изцяло правилно попълнен отговор носи 1 точка.
Отговор: 6

5) Изцяло правилно попълнен отговор носи 1 точка.
Отговор: 8

6) Всеки изцяло коректно попълнен ред носи по 1 точка (общо 3 точки).

```
void insert(int* arr, int index) {
  int key = arr[index];
  int j = index - 1;

  while (j >= 0 && arr[j] > key) {
    arr[j + 1] = arr[j];
    j = j - 1;
  }
  arr[j + 1] = key;
}
```

7) Всяко изцяло коректно попълнено извикване на функцията `std::reverse` носи по 1 точка (общо 2 точки).

```
void rotateArray(int* nums,
  int k, int size) {
  if (size == 0 || k % size == 0)
    return;
  k %= size;
  std::reverse(nums, nums+size);
  std::reverse(nums, nums+k);
  std::reverse(nums+k, nums+size);
}
```

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

Разгледайте дадения по-долу фрагмент. След това отговорете на въпросите, които следват.

```
1  class Base {
2  public:
3    ~Base() {}
4    virtual Base& self() { return *this; }
5  };
6  class Derived : virtual public Base {
7    int* ptr;
8  public:
9    Derived() { ptr = new int[10]; }
10   ~Derived() { delete[] ptr; }
11   Derived& self() { return *this; }
12 };
13 int main() {
14   Base* b = new Derived();
15   delete b;
16 }
```

1) Срещу всяко от твърденията напишете „Да“, ако то е вярно, или „Не“, ако е невярно.

А) Класът `Base` е абстрактен, защото в него има виртуална функция.	
Б) Класът `Base` е полиморфен, защото в него има виртуална функция.	
В) Функцията `Base::self` е чисто виртуална (pure virtual).	
Г) Функцията `Derived::self` е `public`, защото и `Base::self` е `public`.	
Д) `Base::self` връща референция (reference) към временен обект.	
Е) В `Base::self` има грешка, защото връща указател, а типът ѝ е референция – `Base&`.	

2) По-долу са описани няколко наблюдения върху по-горния фрагмент. Срещу всяко от тях напишете „Грешка“, ако то описва проблем в кода или „ОК“, ако описва валидна конструкция.

А) Деструкторът на `Base` не е виртуален, а съдържа виртуална функция и наследникът му заделя динамични ресурси.	
Б) Деструкторът на `Base` е публичен.	
В) `self` е дефинирана като `virtual` в `Base`, но не и в `Derived`.	
Г) Типът на `Base::self` е `Base`, но `Derived::self` има тип `Derived`.	

3) Разгледайте дадения по-долу фрагмент.

```
1  virtual class Foo {
2  public:
3    virtual Foo() {}
4    virtual ~Foo() {}
5    virtual int var;
6    int func(virtual int param) {
7      return param;
8    }
9    virtual void anotherFunc() {
10     virtual int localVar;
11   }
12 };
13
14 class Bar : virtual public Foo { };
```

На кои редове запазената дума `virtual` е използвана коректно? В полетата за отговор по-долу напишете „Да“ срещу всяко място, където употребата е коректна и „Не“, където има грешка.

ред 1, дефиниция на клас	
ред 3, дефиниция на конструктор	
ред 4, дефиниция на деструктор	
ред 5, дефиниция на член-променлива	
ред 6, дефиниция на параметър на функция	
ред 9, дефиниция на член-функция	
ред 10, дефиниция на локална променлива във функция	
ред 14, наследяване на клас	

4) Довършете програмата на подчертаните места, така че да извежда текст „ABC“ на екрана.

```
1  // Обърнете внимание, че функциите
2  // извеждат точно по един символ!
3  class A {
4  public:
5    A() { std::cout << '___'; }
6    ________ void f() { std::cout << '___'; }
7  };
8  class B : public A {
9  public:
10   B() { std::cout << '___'; }
11 };
12 int main() {
13   ___ obj;
14   ___::f();
15 }
```

Примерно решение

1)

А)	Не
Б)	Да
В)	Не
Г)	Не
Д)	Не
Е)	Не

2)

А)	Грешка
Б)	ОК
В)	ОК
Г)	ОК

3)

ред 1, дефиниция на клас	Не
ред 3, дефиниция на конструктор	Не
ред 4, дефиниция на деструктор	Да
ред 5, дефиниция на член-променлива	Не
ред 6, дефиниция на параметър на функция	Не
ред 9, дефиниция на член-функция	Да
ред 10, дефиниция на локална променлива във функция	Не
ред 14, наследяване на клас	Да

4)

```
1  // Обърнете внимание, че функциите
2  // извеждат точно по един символ!
3  class A {
4  public:
5    A() { cout << 'A'; }
6    static void f() { cout << 'C'; }
7  };
8  class B : public A {
9  public:
10   B() { cout << 'B'; }
11 };
12 int main() {
13   B obj;
14   A::f(); // алтернативен вариант е и
       B::f();
15 }
```

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:

1) 3 точки (по 0,5 за всеки верен отговор);

2) 2 точки (по 0,5 за всеки верен отговор);

3) 2 точки (по 0,25 за всеки верен отговор);

4) 3 точки (по 0,5 за всяко вярно попълнено място);

Задача 3. Разглеждаме (кореново) дърво с върхове, чиито стойности са цели числа, като всеки връх може да има произволен брой наследници, редът на които ще има значение – първия ще наричаме най-ляв, а последния – най-десен. „Ниво с номер n“ наричаме списък от върховете в дървото, които са на разстояние n от корена, подредени от ляво надясно. Казваме, че ниво в дървото „представя“ даден вектор от цели числа, ако елементите във върховете в нивото образуват точно елементите на вектора.

А) Да се дефинира тип `TreeNode` за представяне на връх от такова дърво.

Б) Да се реализира функция

```
int findLevel(const TreeNode* root, const vector<int>& numbers);
```

която по подаден указател към корен на такова дърво намира номер на ниво, представящо вектора `numbers`, или -1, ако такова ниво няма. Ако има повече от едно ниво с такива елементи, не е от значение кой номер ще се върне.

Приемаме, че коренът определя ниво с номер 0.

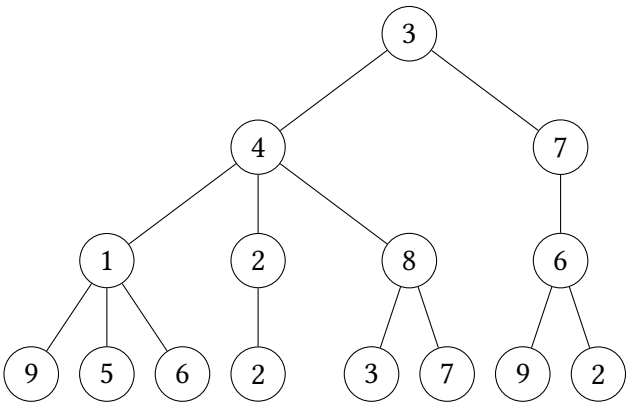
В) Да се реализира функция

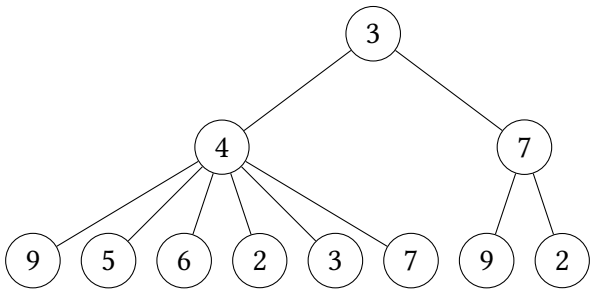
```
void removeAllLevels(TreeNode* root, const vector<int>& numbers)
```

която по подаден указател към корен на дърво премахва всички нива с положителен номер, които представят вектора `numbers`. При премахване на връх, неговите преки наследници заемат неговото място и стават преки наследници на родителя му. Приемаме, че елементите на дървото са заделени с `new`.

Задачата да се реализира на езика C/C++. Позволено е използването на всички средства от стандартната библиотека.

Пример:
За дървото вляво и вектор от числа { 1, 2, 8, 6 }, функцията `findLevel` трябва да върне 2, а след изпълнението на `removeAllLevels` трябва да се получи дървото вдясно.





Примерно решение

Забележка:

Включва допълнителни функции, демонстриращи работата на програмата, които не се изискват в решението на поставената задача.

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

using std::vector;
using std::queue;

// Подточка А
struct TreeNode
{
    int data;
    vector<TreeNode*> successors;
    TreeNode(int x) :data(x) {}
};

// Подточка Б
// намира последното такова ниво за да може да работи добре изтриването,
// ако нивото на корена отговаря на условието
int findLevel(const TreeNode* root, const vector<int>& numbers)
{
    if (!root) return -1;
    queue<const TreeNode*> level;

    int currentLevel = 0;
    int foundLevel = -1;
    level.push(root);
    level.push(nullptr);

    vector<int> levelData;
    while (level.size() > 0) {
        const TreeNode* element = level.front();
        level.pop();
        if (element == nullptr) {
            if (numbers == levelData)
                foundLevel = currentLevel;
            levelData.clear();
            ++currentLevel;
            if (!level.empty()) level.push(element);
        }
        else {
            levelData.push_back(element->data);
            for (const TreeNode* successor : element->successors)
                level.push(successor);
        }
    }
    return foundLevel;
}

// Подточка В
```

```
void removeAtLevel(TreeNode* root, int level) {
    if (!root || level < 1) return;

    if (level == 1) {
        vector<TreeNode*> children(std::move(root->successors));
        root->successors.clear();
        for (TreeNode* child : children) {
            for (TreeNode* grandchild : child->successors)
                root->successors.push_back(grandchild);
            delete child;
        }
    }
    else {
        for (TreeNode* child : root->successors)
            removeAtLevel(child, level - 1);
    }
}

void removeAllLevels(TreeNode* root, const vector<int>& numbers)
{
    for (int level = findLevel(root, numbers); level >= 1; level = findLevel(root, numbers))
        removeAtLevel(root, level);
}

// Следват допълнителни функции за демонстрация, извън задачата
void print(const TreeNode* root)
{
    std::cout << '\n';
    if (!root) return;
    queue<const TreeNode*> queue;
    queue.push(root);
    while (!queue.empty()) {
        int cnt = (int)queue.size();
        for (int i = 0; i < cnt; ++i) {
            const TreeNode* elem = queue.front();
            queue.pop();
            std::cout << elem->data << ' ';
            for (const TreeNode* child : elem->successors)
                queue.push(child);
        }
        std::cout << '\n';
    }
    std::cout << '\n';
}

void clear(TreeNode* root)
{
    if (root) {
        for (TreeNode* node : root->successors)
            clear(node);
        delete root;
    }
}
```

```
int main()
{
    TreeNode* root = new TreeNode(3),
        * e1 = new TreeNode(4), * e2 = new TreeNode(7);
    root->successors.push_back(e1);
    root->successors.push_back(e2);

    TreeNode* e3 = new TreeNode(1), * e4 = new TreeNode(2), * e5 = new TreeNode(8);
    e1->successors.push_back(e3);
    e1->successors.push_back(e4);
    e1->successors.push_back(e5);

    TreeNode* e6 = new TreeNode(6);
    e2->successors.push_back(e6);

    e3->successors.push_back(new TreeNode(9));
    e3->successors.push_back(new TreeNode(5));
    e3->successors.push_back(new TreeNode(6));
    e4->successors.push_back(new TreeNode(12));
    e5->successors.push_back(new TreeNode(13));
    e5->successors.push_back(new TreeNode(17));
    e6->successors.push_back(new TreeNode(9));
    e6->successors.push_back(new TreeNode(12));

    print(root);

    std::cout << findLevel(root, { 3 }) << '\n';
    std::cout << findLevel(root, { 4, 7 }) << '\n';
    std::cout << findLevel(root, { 2, 8, 6 }) << '\n';
    std::cout << findLevel(root, { 1, 2, 8, 6 }) << '\n';
    std::cout << findLevel(root, { 9, 5, 6, 12, 13, 17, 9, 12}) << '\n';
    std::cout << findLevel(root, { 3, 4, 6, 23 }) << '\n';

    removeAllLevels(root, { 1, 2, 8, 6 });
    print(root);

    clear(root);

    //-------------------------
    root = new TreeNode(2);
    e1 =  new TreeNode(2);
    e2 =  new TreeNode(2);
    e1->successors.push_back(e2);
    root->successors.push_back(e1);

    print(root);
    cout << findLevel(root, { 2 }) << '\n';

    removeAllLevels(root, { 2 });
    print(root);
    clear(root);

    return 0;
}
```

Критерии за оценяване

А) За вярно дефинирана структура с данни `int` или подобен, която позволява произволен брой наследници се присъжда 1 точка. Във всеки друг случай не се присъждат точки.

Б) За вярно дефинирана функция, която коректно намира елементите от едно ниво и проверява дали точно отговарят на подадените числа, се присъждат 5 точки. Подобно решение може да е реализирано както с обхождане в широчина, така и чрез обхождане в дълбочина. Приемливо е да се генерира последователност от елементите от цяло ниво и да се сравнява с подадения вектор или да се изследват последователно елементите от ниво като се следи съвпадение с подадените числа; при откриване на разлика да се прекратява изследването и да се преминава към следващо ниво.

За пропуснат граничен случай на празно дърво се отнема 1 точка.

За некоректно отчитане на край на ниво се отнемат 2 точки. Такъв пример е да се следи за коректна последователност от върхове във фронта на обхождане, но не се отчита дали са от едно ниво и дали са цяло ниво.

За некоректно отчитане на реда на елементите (например се сравняват на обратно или в произволен ред) се отнемат 2 точки.

За некоректно върнат резултат се отнема 1 точка. Това може да е, например, ако не се връща -1, когато не се открие ниво с исканото свойство, или не се преброяват коректно нивата.

В) За вярно дефинирана функция, която премахва всички съответни нива и коректно освобождава паметта, се присъждат 4 точки. Предполага се да се използва предната подточка и да се премахват всички елементи с подадена дълбочина или частично да се копира логиката от предната подточка за определяне на елементите, формиращи едно ниво.

За коректно премахване само на едно ниво вместо на всички нива с исканото свойство се присъждат 3 точки.

За некоректно освобождаване на паметта за премахнатите елементи (липсва `delete`) се отнема 1 точка.

За некоректно преброяване на нивата (коренът е на ниво 0) се отнема 1 точка.

За некоректно определяне на границите на ниво се отнемат 2 точки.

За некоректна работа с наследниците на премахнат елемент се отнемат 2 точки. Такъв случай е например ако те се губят или не се закачат на правилния родител.

Задача 4.

А) Разглеждаме банкова сметка, която предлага променливи лихвени проценти: 0,5 процента за първите 1 000 лв. кредит, 1 процент за следващите 1 000 лв. кредит и 1,5 процента за останалата сума. Какви валидни класове на еквивалентност биха могли да се дефинират и използват, за да се провери дали банката обработва правилно дадена сметка?

Б) Компания за онлайн поръчки таксува 3,95 лв. за опаковане и транспорт на всички поръчки на стойност до 20 лв. и 4,95 лв. за поръчки на стойност между 20 лв. и 40 лв. За поръчки на стойност над 40 лв. няма такса за разходи за опаковане и транспорт. Да се дефинират класовете на еквивалентност, с които да се тества описаната функционалност.

В) Система е проектирана да приема резултати от независими оценяващи, които могат да оценяват едновременно един и същ изпитен тест. Всеки изпитен тест трябва да има 5 въпроса, които могат да получат максимално 20 т., като се изчислява и обща оценка за целия тест. Резултатите от всеки двама оценяващи се сравняват по отношение на разликите. При наличие на разлика по-голяма от 3 т. за който и да е от въпросите или обща разлика за целия тест по-голяма от 10 т. се извършва повторно оценяване.

1) Да се дефинират класовете на еквивалентност, с които да се тества функционалността на системата.
2) Да се дефинират граничните стойности, с които да се тества функционалността на системата.
3) Да се обоснове дали е възможно редуциране на сценариите, свързани с тестване на граничните стойности.

Примерно решение

А) Класовете на еквивалентност са: $0.00–1\,000,00$ лв., $1\,000,01–2\,000,00$ лв. и $\geq 2\,000,01$ лв.

Б) Валидни класове на еквивалентност са: 0,00–20,00 лв., 20,01–40,00 лв., и $\geq$ 40,01 лв. Невалидни класове на еквивалентност могат включват отрицателни стойности и символи.

В) Класовете на еквивалентност са: оценка на въпрос от тест 0–20 т., обща оценка на тест 0–100 т.; разлика в оценка на въпрос: 0–3 т. и $>$ 3 т.; разлика в обща оценка на теста 0–10 т. и $>$ 10 т. Граничните стойности са: -1, 0, 1 и 19, 20, 21 за оценка на въпросите; -1, 0, 1 (отново) и 99, 100, 101 за обща оценка на теста; -1, 0, 1 (отново) и 2, 3, 4 за разлика в оценка на въпрос; и -1, 0, 1 (отново) и 9, 10, 11 за разлика в обща оценка на теста. Стойностите -1, 0, 1 се появяват няколко пъти, но тъй като могат да бъдат приложени върху различни части на системата, е необходимо тяхното повторение в различните сценарии.

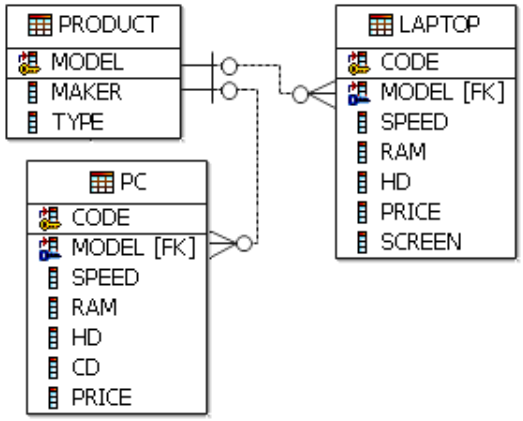
Критерии за оценяване

А) 2 т.

Б) 2 т.

В) 1) 2 т.
2) 3 т.
3) 1 т.

Задача 5. Дадена е базата от данни PC. В нея се съхранява информация за два вида продукти – лаптопи и персонални компютри.



Таблицата Product съдържа базова информация за всеки продукт:

- model — модел на продукта, първичен ключ;
- maker — производител на продукта;
- type — един от типовете: 'PC', 'Laptop'.

Таблицата PC съдържа специфична информация за персоналните компютри:

- code — уникален идентификатор на дадена компютърна конфигурация, първичен ключ;
- model — модел на компютъра, външен ключ към Product.model. Може да имаме няколко различни компютърни конфигурации от един и същ модел, но с различни параметри. Може да бъде неизвестен (NULL);
- speed — тактова честота на процесора в GHz;
- ram — количество RAM памет в GB;
- hd — размер на твърдия диск в TB;
- cd — скорост на компактдисковото устройство;
- price — цена на компютъра.

Таблицата Laptop съдържа специфична информация за лаптопите. Атрибутите са аналогични на тези на PC, но липсва атрибутът cd и има атрибут screen, указващ диагонала на екрана в инчове.

1) Да се напише заявка, която извежда кодовете, цените и имената на производителите на всеки персонален компютър, който удовлетворява някое от следните условия:

- моделът започва с буквата „A“ и завършва с цифрата 2;
- моделът е неизвестен (NULL в колоната model на таблицата PC).

За компютърни конфигурации с неизвестен модел трябва да се изведе NULL в колоната за производител.

2) Да се посочи заявката, която извежда без повторение имената на всички производители, които имат едновременно поне един модел персонален компютър и поне два модела лаптопи:

```
А) (SELECT maker
    FROM Product
    WHERE type = 'PC'
      AND COUNT(model) >= 1)
    INTERSECT
    (SELECT maker
    FROM Product
    WHERE type = 'Laptop'
      AND COUNT(model) >= 2);
```

```
Б) SELECT DISTINCT maker
   FROM Product
   WHERE type = 'PC'
     AND maker = (SELECT maker
       FROM Product
       GROUP BY maker
       HAVING COUNT(maker) >= 2
         AND type = 'Laptop');
```

```
В) SELECT maker
   FROM Product p
   WHERE type = 'Laptop'
     AND EXISTS (SELECT *
                 FROM Product
                 WHERE type = 'PC'
                   AND maker = p.maker)
   GROUP BY maker
   HAVING COUNT(*) >= 2;
```

```
Г) (SELECT maker
    FROM Product
    WHERE type = 'Laptop'
    GROUP BY maker
    HAVING COUNT(*) >= 2)
    EXCEPT
    (SELECT maker
    FROM Product
    WHERE type != 'PC');
```

Примерно решение на подзадача 1:

```
SELECT code, price, maker
FROM PC
LEFT JOIN Product ON PC.model = Product.model
WHERE PC.model LIKE 'A%2' OR PC.model IS NULL;
```

Критерии за оценяване:

1) Коректно решение носи 5 т. За всяка сгрешена клауза броят на точките се намалява с 2 до достигане на 0 т.

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.

Да се направи декомпозиция на модулите от архитектурата на IoT софтуерно приложение за *интелигентна домашна сигурност (smart home security system).* Да се обоснове защо така проектираната архитектура удовлетворява изискванията.

R1. Потребителите трябва да могат да влизат сигурно в системата, като използват идентификационни данни като потребителско име и парола. Приложението има два вида потребителски профили:

- обикновени потребители, които използват системата;
- администратори, които следят за изправността на системата.

R2. Потребителският интерфейс трябва да бъде интуитивен и удобен за потребителя, обслужващ потребители с различен технически опит, за да се гарантира лесна употреба.

R3. Приложението трябва да поддържа интеграция с различни IoT устройства в дома, като сензори за движение, сензори за врати и прозорци, охранителни камери и интелигентни ключалки.

R4. Приложението трябва да бъде проектирано да скалира безпроблемно с добавянето на повече устройства към интелигентната домашна система без компромис с производителността.

R5. Приложението трябва да изпраща предупреждения до потребителите чрез насочени известия или имейли в случай на пробиви в сигурността, като например неупълномощено влизане в дома, открито от сензорите.

R6. Потребителите трябва да могат да контролират дистанционно свързани устройства, като активиране/деактивиране на системата за сигурност, заключване/отключване на врати и включване/изключване на светлини.

R7. Потребителите трябва да могат да наблюдават състоянието на дома си в реално време чрез приложението, включително състоянието на сензорите и камерите.

R8. Приложението трябва да поддържа журнални (log) файлове и статистики на всички дейности, включително активиране на сензори, контроли на устройството и потребителски вход, достъпни за преглед от потребителите.

R9. Системата трябва да има висока надеждност и не трябва да има откази, като максималното време в което системата не е достъпна да е до 4 часа годишно.

R10. Приложението трябва да гарантира, че потребителските данни и самата интелигентна домашна система са защитени срещу неупълномощен достъп.

Примерно решение

Декомпозицията на модулите на системата е представена на фиг. 1. Според условието на задачата, системата е разделена на две подсистеми – сървърна част (Server) и инсталираните при клиента устройства (Home Devices). Потребителският интерфейс на системата е реализиран в модула UI.

Модул User Handling отговаря за функционалностите относно достъпа на потребителите. Предоставя два под-модула, Register и Login, чрез които се манипулират потребителските профили. Чрез тях се осъществяват регистрацията на потребителите и удостоверяването и упълномощаването (роли и т.н.) при влизане в системата. Те комуникират с базата данни, където е съхранена информацията, която е криптирана. С цел сигурност, освен криптирането на данните, всички комуникации са криптирани, използвайки сигурни протоколи. Също така, тук се намира и, евентуално, допълнителна логика, свързана с правата на всеки потребител и достъпа до системата. По този начин са изпълнени изискванията R1 и R10.

Отдалеченото контролиране на устройствата се осъществява чрез модул Control, където се обработват командите на потребителя. По същия начин наблюдението в реално време и преглеждането на различни журнали и статистики се поддържа, съответно, от модулите Monitoring и Logs. Тези три под-модула, които се грижат за основната функционалност относно взаимодействията с потребителя, са части от модула Home Manager, с който се удовлетворяват изискванията R6, R7 и R8.

В модул Alert се осъществява изпращането на предупрежденията до потребителите въз основа на предварително зададени тригери, и така е удовлетворено изискване R5.

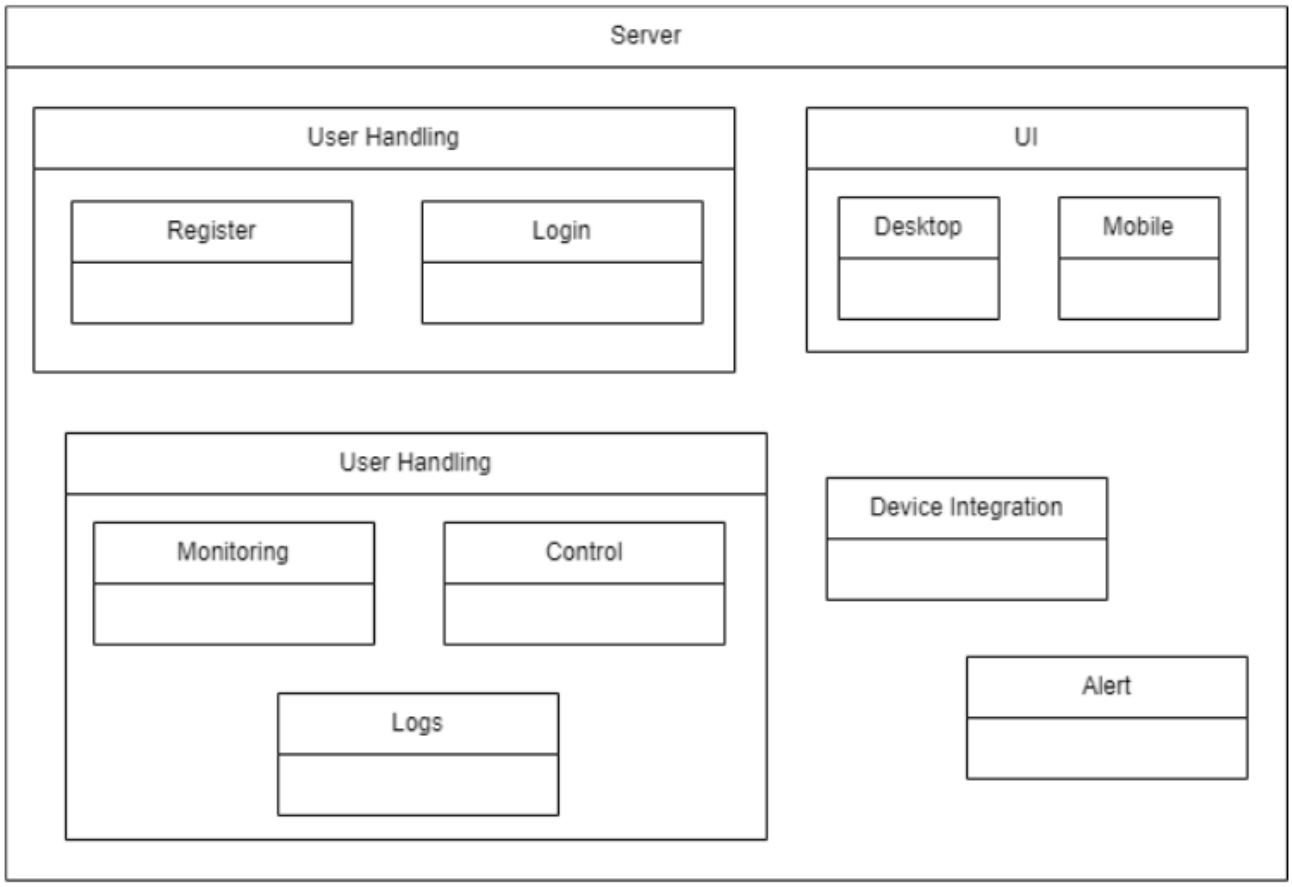
Home Devices модулът представлява полевата част на системата при клиента. Модулът Devices съдържа различните устройства от всеки дом – Sensors, Cameras, Locks – като в него може да се добавят и други типове. Чрез модула Communication module се осъществява комуникацията и интеграцията на устройствата със системата, изпълнявайки изискване R3.

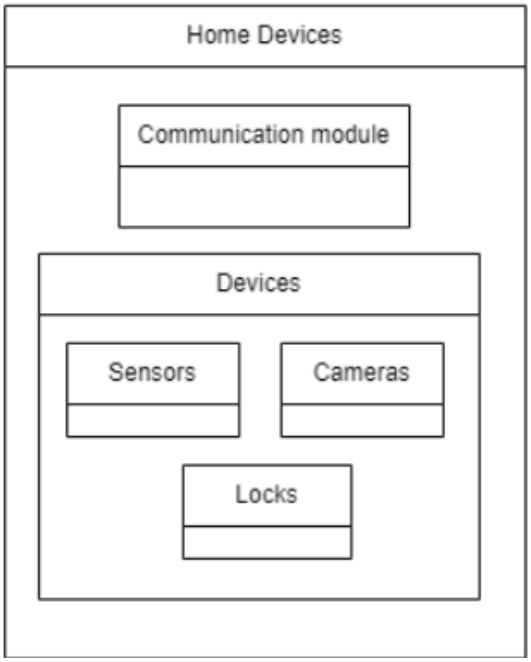
Модул Device Integration, при нарастващ брой устройства, улеснява тяхното добавяне към системата, при нужно скалиране. Осигурява инсталирането и се грижи за успешната комуникация между новите устройства и системата и удовлетворява изискване R4.

Чрез репликация на сървъра и базата данни и използването на подходящи процеси (напр. Load balancing и др.) се осигуряват изискванията относно качествените атрибути за надеждност и производителност (изискване R9) (*репликацията не е показана на фиг. 1, тъй като се счита, че тя не е предмет на декомпозицията на модулите*).

Критерии за оценяване

Удовлетворено е изискването за потребителски интерфейс (R2)	1 т.
Удовлетворено е изискването за регистрация, удостоверяване и сигурност на потребителите и данните (R1, R10)	1 т.
Удовлетворени са изискванията за контролиране и наблюдаване (R6, R7)	2 т.
Удовлетворено е изискването за интеграция на различните устройства (R3)	2 т.
Удовлетворено е изискването за преглеждане на журнали и статистики (R8)	1 т.
Удовлетворено е изискването за изпращане на предупреждения (R5)	1 т.
Удовлетворено е изискването за добавяне на нови устройства (R4)	1 т.
Удовлетворени са изискванията за надеждност (reliability) (R9)	1 т.





Фигура 1: Декомпозиция на модулите

Задача 7. Да се пресметне интегралът

$$\int_{\frac{1}{\sqrt{3}}}^{\sqrt{3}} \operatorname{arctg}\left(\frac{1}{x}\right) \mathrm{d}x$$

Примерно решение и критерии за оценяване

Интеграла намираме чрез интегриране по части:

$$\int_{\frac{1}{\sqrt{3}}}^{\sqrt{3}} \operatorname{arctg}\left(\frac{1}{x}\right) \mathrm{d}x \;=$$

(2 т.)
$$= x \operatorname{arctg}\left(\frac{1}{x}\right)\Big|_{\frac{1}{\sqrt{3}}}^{\sqrt{3}} - \int_{\frac{1}{\sqrt{3}}}^{\sqrt{3}} x \,\mathrm{d}\left[\operatorname{arctg}\left(\frac{1}{x}\right)\right] =$$

(1 т. + 1 т.)
$$= \sqrt{3}\operatorname{arctg}\left(\frac{1}{\sqrt{3}}\right) - \frac{1}{\sqrt{3}}\operatorname{arctg}(\sqrt{3}) - \int_{\frac{1}{\sqrt{3}}}^{\sqrt{3}} x \cdot \frac{-1}{x^2} \cdot \frac{1}{1+\frac{1}{x^2}}\,\mathrm{d}x =$$

(1 т. + 1 т.)
$$= \sqrt{3}\cdot\frac{\pi}{6} - \frac{1}{\sqrt{3}}\cdot\frac{\pi}{3} + \int_{\frac{1}{\sqrt{3}}}^{\sqrt{3}} \frac{x}{x^2+1}\,\mathrm{d}x =$$

(1 т. + 1 т.)
$$= \frac{\pi\sqrt{3}}{18} + \frac{1}{2}\ln(x^2+1)\Big|_{\frac{1}{\sqrt{3}}}^{\sqrt{3}} =$$

(0 т. + 1 т.)
$$= \frac{\pi\sqrt{3}}{18} + \frac{1}{2}\ln(3+1) - \frac{1}{2}\ln(\frac{1}{3}+1) =$$

(0 т. + 1 т.)
$$= \frac{\pi\sqrt{3}}{18} + \frac{1}{2}\ln(3)$$

Задача 8. Хвърля се зар, на който на 2 от страните му е изписана цифрата 1, на други две – цифрата 3, на една от оставащите две страни – цифрата 5, а на последната страна – цифрата 7. Нека X е случайната величина „цифрата, която е изписана на горната страна на зара при неговото хвърляне“.

а) Да се намери разпределението на X.

б) Да се намерят математическото очакване на дисперсията на X.

в) Да се намери p – вероятността да се падне стойност 1 или 5 при хвърляне на зара. Разглеждаме като „успех“ да се падне стойност 1 или 5 при хвърляне на зара и извършваме 10 хвърляния на зара. Нека Y е броят на „успехите“ при описания експеримент. Какво е разпределението на Y? Да се намери вероятността Y да е по-малко или равно на 1.

Примерно решение

а) Разпределението на случайната величина X е:

$X = x$ (стойност)	1	3	5	7
$P(X = x)$ (вероятност)	2/6	2/6	1/6	1/6

б)

$$\begin{aligned} EX &= \sum_x xP(X = x) = 10/3 \approx 3.33 \\ E(X^2) &= \sum_x x^2. \\ P(X = x) &= 94/6 \approx 15.67 \\ VarX &= E(X - EX)^2 = E(X^2) - (EX)^2 \approx 4.56 \end{aligned}$$

в) $p = P(X = 1) + P(X = 5) = 1/2$. Разпределението на Y е биномно със следните параметри: $Y \sim Bi(n = 10, p = 1/2)$ или $P(Y = k) = \binom{10}{k}(0.5)^{10}, k = 1, 2, \ldots, 10; P(Y \leq 1) = (1-p)^{10} + \binom{10}{1}p^1(1-p)^9 \approx 0.0107$.

Критерии за оценяване

- Пълно решение на подточка а) – 4 точки, за всяка стойност на случайната величина и нейната вероятност по 1 т.;
- Пълно решение на подточка б) – 2 точки, за намирането на математическото очакване и дисперсията по 1 т.;
- Пълно решение на подточка в) – 4 точки, за намирането на p – 1 точка, за намирането на разпределението на Y – 2 точки, за намирането на $P(Y \leq 1)$ – 1 точка.

Чернова