

СОФИЙСКИ УНИВЕРСИТЕТ
“СВ. КЛИМЕНТ ОХРИДСКИ”



ФАКУЛТЕТ ПО МАТЕМАТИКА
И ИНФОРМАТИКА

ДЪРЖАВЕН ИЗПИТ

ЗА ПОЛУЧАВАНЕ НА ОКС “БАКАЛАВЪР ПО ИНФОРМАТИКА”

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;
	}

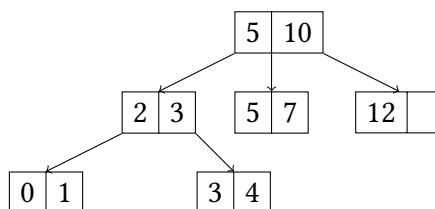
Задача 3. Троично дърво ще наричаме наредено дърво, за което всеки възел съхранява една или две стойности и има не повече от три наследника. Ако за произволен възел означим стойностите на двата елемента в него с X и Y , а трите негови потенциални поддървета с $STleft$, $STmid$ и $STright$, то за да е валидно такова дърво, трябва да са в сила следните свойства:

- стойността X трябва да е строго по-малка от стойността Y ;
- стойността на всеки елемент от поддървото $STleft$ трябва да е строго по-малка от X ;
- стойността на всеки елемент от поддървото $STmid$ трябва да е по-голяма или равна на X и строго по-малка от Y ;
- стойността на всеки елемент от поддървото $STright$ трябва да е по-голяма или равна на Y ;
- ако в даден възел няма втора стойност (Y), то тогава поддървото $STright$ задължително трябва да е празно;
- ако в поддървото $STmid$ на даден възел има поне един елемент по-голям от X , то във възела задължително трябва да има две стойности (X и Y);
- всяко от $STleft$, $STmid$ и $STright$ или е празно, или е валидно троично дърво.

Така неформално можем да напишем следното неравенство:

$$STleft < X \leq STmid < Y \leq STright$$

Пример за такова дърво е показан на фигурата:



- 1) Да се опише структура, с която може да се представи такова троично дърво с елементи цели числа и да се напише нейната дефиниция на C++.
- 2) Да се дефинира функция `insert` с аргументи троично дърво от цели числа и цяло число, която добавя това число в дървото, съблюдавайки изискванията за валидност. При добавяне на нова стойност не трябва да се променят или преместват съществуващите в дървото стойности.
- 3) Да се дефинира функция `find` с аргументи троично дърво от цели числа и цяло число, която проверява дали подаденото число се съдържа в дървото.
- 4) Да се реализира кратка програма, която демонстрира описаните функции, като за целта:
 - а) създава празно дърво;
 - б) въвежда от стандартния вход естествено число N , последвано от N цели числа, които добавя в дървото;
 - в) прочита от стандартния вход едно число и извежда на стандартния изход подходящ текст, указващ дали това число се среща или не в дървото;
 - г) подсигурява коректна работа с паметта.

Задачата трябва да се реализира на езика C++. Позволено е използването на класовете и методите от стандартната библиотека, отговаряща на стандарта C++14 или по-стар.

Примерно решение

```
#include <iostream>

struct node
{
    int X, Y;
    node* left  = nullptr,
        * mid   = nullptr,
        * right = nullptr;
};

struct tree
{
    node* root = nullptr;
};

void insert(node*& root, int value)
{
    if (!root) {
        root = new node;
        root->X = root->Y = value;
        return;
    }

    if (value < root->X) insert(root->left, value);
    else if (value == root->X) insert(root->mid, value);
    else if (root->X == root->Y) root->Y = value;
    else if (value < root->Y) insert(root->mid, value);
    else insert(root->right, value);
}

void insert(tree& t, int value)
{
    insert(t.root, value);
}

bool find(const node* root, int value)
{
    if (!root) return false;
    if (value == root->X || value == root->Y) return true;
    if (value < root->X) return find(root->left, value);
    if (value < root->Y) return find(root->mid, value);
    return find(root->right, value);
}

bool find(const tree& t, int value)
{
    return find(t.root, value);
}

void clear(node* root)
{
    if (root) {
```

```
        clear(root->left);
        clear(root->mid);
        clear(root->right);
        delete root;
    }
}
void clear(tree& t)
{
    clear(t.root);
    t.root = nullptr;
}

int main()
{
    tree t;

    int N;
    do {
        std::cout << "N = ";
        std::cin >> N;
    } while (N <= 0);
    for (int i = 0; i < N; ++i) {
        int value;
        std::cin >> value;
        insert(t, value);
    }

    int X;
    std::cin >> X;
    std::cout << "The tree "
              << (find(t, X) ? "contains " : "does not contain ")
              << X << "\n";

    clear(t);
    return 0;
}
```

Критерии за оценяване

- 1) За коректно дефинирана структура с две полета цели числа и три указателя и посочване на дървото като корен (указател към тази структура) – 1 т.
 - Допуска се и шаблон на такава структура.
 - Не се изисква дефиниране на клас Tree, но трябва по някакъв начин явно да се укаже, че дървото се представя чрез корена си.
- 2) За коректно дефинирана функция insert – 4 т.
- 3) За всеки от петте случая на вмъкване (X, Y, STleft, STmid, STright) – по 0,5 точки.
- 4) За коректно променяне на подадения указател (поне за вмъкване в празно дърво) 0,5 точки.
- 5) За коректно справяне със случая с една стойност във възел – 1 т.
- 6) За коректна функция find – 3 т.

- 7) За случая с празно дърво – 0,5 т.
- 8) За намиране на данните в текущия възел – 1 т.
- 9) За коректно слизване по трите поддървета – 1,5 т. (по 0,5 т. за всеки случай)
 - а) Не се присъждат точки за създаване на празно дърво.
 - б) За коректно четене на N, последвано от N числа и използване на функцията от 2) – 0,5 т.
 - в) За коректно използване на функцията от 3) и обработка на резултата от нея – 0,5 т.
 - г) За коректна функция, която да изчисти паметта за дървото – 1 т. Допуска се имплементация на деструктор, както и друга техника, чрез която да се освободи заетата памет.

Сумата от точките се закръгля до цяло число.

Задача 4. Задачата да се реши на един от езиците Scheme или Haskell.

Обществото на информатиците провежда избори за съвет на старейшините. „Кандидатура“ е наредена двойка от низове: името на кандидата и неговата специалност, а „кандидатската листа“ е списък от кандидатури. Може да се гласува за произволен брой кандидати. Попълнените „бюлетини“ представляват предикати, които приемат кандидатура и връщат истина или лъжа, в зависимост дали се гласува за съответната кандидатура или не. Да се попълнят празните полета по-долу така, че:

- votes да връща броя на гласовете от бюлетините ballots за кандидата с име cand;
- election да връща списък от наредени двойки от името на кандидата и броя от гласовете за него от бюлетините ballots в реда, в който кандидатите се срещат в листата cl;
- sortResults да сортира резултатите, върнати от функцията election, в низходящ ред по броя на гласовете, като кандидатите с равен брой гласове запазват реда си от листата;
- selectCouncil да връща списък от имена от избраните **не повече** от max кандидати, които имат повече от половината гласове, в реда, върнат от sortResults.

Упътване: могат да се използват наготово всички функции в R5RS за Scheme и в Prelude за Haskell.

Haskell

```
votes cand ballots = _____
election ballots cl = _____ (\_____ -> (_____, _____)) cl
sortResults [] = []
sortResults _____ = more ++ equal ++ less
  where more = _____
        equal = _____
        less = _____
selectCouncil max ballots cl =
  _____ (map _____ (filter _____ results))
  where results = sortResults (election ballots cl)
```

Пример:

```
cl = [("Kernighan","C"),("Ritchie","C"),("Stroustrup","C++"),("Steele","Scheme"),
      ("Sussman","Scheme"),("Church","Lambda"),("Curry","Lambda")]
b1 (name, specialty) = specialty == "Lambda" || last name == 'e'
b2 (name, specialty) = name == "Church" || head specialty == 'C'
b3 (name, specialty) = length name > 6 && specialty /= "C++"
selectCouncil 2 [b1, b2, b3] cl = ["Ritchie","Kernighan"]
```

Scheme

```
(define (votes cand ballots) _____)
(define (election ballots cl) (_____ (lambda (c) (cons _____ _____)) cl))
(define (sortResults res) (if (null? res) '()
  (let ((more _____)
        (equal _____)
        (less _____))
    (append more equal less))))
(define (selectCouncil max ballots cl)
  (define (select max results) (if (or (null? results) _____) '()
    (cons _____ (select (- max 1) (cdr results)))))
  (select max (sortResults (election ballots cl))))
```

Пример:

```
(define cl '(("Kernighan"."C") ("Ritchie"."C") ("Stroustrup"."C++") ("Steele"."Scheme")
  ("Sussman"."Scheme") ("Church"."Lambda") ("Curry"."Lambda")))
(define (b1 c) (or (equal? (cdr c) "Lambda") (eqv? (car (reverse (string->list (car c)))) #\e)))
(define (b2 c) (or (equal? (car c) "Church") (eqv? (string-ref (cdr c) 0) #\C)))
(define (b3 c) (and (> (string-length (car c)) 6) (not (equal? (cdr c) "C++"))))
(selectCouncil 2 (list b1 b2 b3) cl) → ("Ritchie" "Kernighan")
```

Примерни решения

Haskell

```
votes cand ballots = length $ filter ($ cand) ballots

election ballots cl = map (\cand@(name, _) -> (name, votes cand ballots)) cl

sortResults [] = []
sortResults ((name,votes):results) = more ++ equal ++ less
  where more = sortResults $ filter (\(_,v) -> v > votes) results
        equal = (name, votes) : filter (\(_,v) -> v == votes) results
        less = sortResults $ filter (\(_,v) -> v < votes) results

selectCouncil max ballots cl =
  take max (map fst (filter ((> length ballots `div` 2) . snd) results))
  where results = sortResults (election ballots cl)
```

Scheme

```
(define (votes cand ballots) (length (filter (lambda (b) (b cand)) ballots)))

(define (election ballots cl) (map (lambda (c) (cons (car c) (votes c ballots))) cl))

(define (sortResults res) (if (null? res) '()
  (let ((more (sortResults (filter (lambda (c) (> (cdr c) (cdar res))) res)))
        (equal (filter (lambda (c) (= (cdr c) (cdar res))) res))
        (less (sortResults (filter (lambda (c) (< (cdr c) (cdar res))) res))))
    (append more equal less))))

(define (selectCouncil max ballots cl)
  (define (select max results)
    (if (or (null? results) (= max 0) (<= (cdar results) (/ (length ballots) 2))) '()
        (cons (caar results) (select (- max 1) (cdr results)))))
  (select max (sortResults (election ballots cl))))
```

Критерии за оценяване

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

Haskell

По ред на празните слотове:

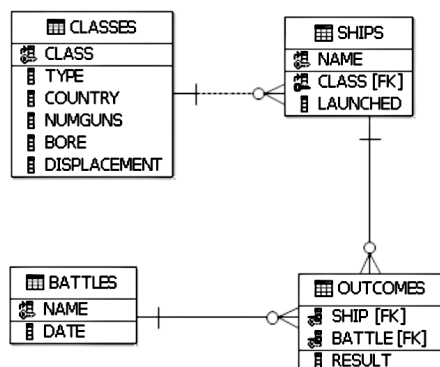
1. **1,5 т.** за функцията `length $ filter ($ cand) ballots` или еквивалентна на нея, която връща броя на елементите на списъка `ballots`, които връщат `True` след прилагане над `cand`;
 - **1 т.** за функция, която третира `ballots` като списък от имена на кандидати или списък от кандидатури и коректно намира броя на тези от тях, за които името на кандидата съвпада с `cand`;
2. **0,5 т.** за функцията `map`;
3. не се дават или отнемат точки, служи само за именуване на променливите, които се използват в другите слотове;
4. **0,5 т.** за първия компонент на параметъра от слот 3;
5. **1 т.** за извикването на функцията `votes` над параметъра от слот 3 и параметъра `ballots`;
 - признават се и решения, в които `votes` се извиква над първия компонент на параметъра от слот 3 и параметъра `ballots`;
6. не се дават или отнемат точки, служи само за именуване на променливите, които се използват в другите слотове;
7. **1,5 т.** за рекурсивно извикване на функцията `sortResults` над тези от елементите на параметъра от слот 6, чиито втори компонент е **по-голям** от втория компонент на главата на параметъра от слот 6;
 - **1 т.** при коректна реализация, която нарушава първоначалния ред на елементите, неправилно достъпва компонентите при сравнение или пропуска рекурсивното извикване на `sortResults`;
8. **1,5 т.** за тези от елементите на параметъра от слот 6, чиито втори компонент е **равен** на втория компонент на главата на параметъра от слот 6;
 - **1 т.** при коректна реализация, която нарушава първоначалния ред на елементите, неправилно достъпва компонентите при сравнение, прави рекурсивно извикване към `sortResults` или пропуска първия елемент на параметъра от слот 6;
9. **1,5 т.** за рекурсивно извикване на функцията `sortResults` над тези от елементите на параметъра от слот 6, чиито втори компонент е **по-малък** от втория компонент на главата на параметъра от слот 6;
 - **1 т.** при коректна реализация, която нарушава първоначалния ред на елементите, неправилно достъпва компонентите при сравнение или пропуска рекурсивното извикване на `sortResults`;
10. **0,5 т.** за `take max` или еквивалентен израз, който извлича първите `max` елемента от списък;
11. **0,5 т.** за `fst`;
12. **1 т.** за предикат, който връща `True` точно за наредени двойки, чиито втори компонент надвишава половината на дължината на `ballots`;
 - признава се и предикат, който връща `True` точно за наредени двойки, чиито втори компонент надвишава половината от общия брой на всички получени гласове за всички кандидати;
 - **0,5 т.** при грешка в типовете при деление на две (например, липса на `fromIntegral` и използване на `/` вместо `div`).

Scheme

По ред на празните слотове:

1. **1,5 т.** за функцията `(length (filter (lambda (b) (b cand)) ballots))` или еквивалентна на нея, която връща броя на елементите на списъка `ballots`, които връщат `#t` след прилагане над `cand`;
 - **1 т.** за функция, която третира `ballots` като списък от имена на кандидати или списък от кандидатури и коректно намира броя на тези от тях, за които името на кандидата съвпада с `cand`;
2. **0,5 т.** за функцията `map`;
3. **0,5 т.** за `(car c)`;
4. **1 т.** за `(votes c ballots)`
 - признава се също и `(votes (car c) ballots)`;
5. **1,5 т.** за рекурсивно извикване на функцията `sortResults` над тези от елементите на `res`, чиито втори компонент е **по-голям** от `(cdar res)`;
 - **1 т.** при коректна реализация, която нарушава първоначалния ред на елементите, неправилно достъпва компонентите при сравнение или пропуска рекурсивното извикване на `sortResults`;
6. **1,5 т.** за тези от елементите на `res`, чиито втори компонент е **равен** на `(cdar res)`;
 - **1 т.** при коректна реализация, която нарушава първоначалния ред на елементите, неправилно достъпва компонентите при сравнение, прави рекурсивно извикване на `sortResults` или пропуска първия елемент на `res`;
7. **1,5 т.** за рекурсивно извикване на функцията `sortResults` над тези от елементите на `res`, чиито втори компонент е **по-малък** от `(cdar res)`;
 - **1 т.** при коректна реализация, която нарушава първоначалния ред на елементите, неправилно достъпва компонентите при сравнение или пропуска рекурсивното извикване на `sortResults`;
8. очакват се два изрази, записани в произволен ред:
 - **0,5 т.** за `(= max 0)` или еквивалентен на него израз;
 - **1 т.** за израз, проверяващ дали `(cdar results)` **не надвишава** половината от дължината на списъка `ballots`;
 - признава се и израз, който проверява дали `(cdar results)` **не надвишава** половината от общия брой на всички получени гласове за всички кандидати;
9. **0,5 т.** за `(caar results)`.

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 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.

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. Колко низа с дължина n над азбуката $\{a, b, c, d\}$ съдържат поне една буква a , поне една буква b и поне една буква c ?

Отговорът носи точки само ако е придружен от коректна обосновка.

Примерно решение

Нека U е универсалното множество в тази задача: множеството от всички низове с дължина n над азбуката $\{a, b, c, d\}$. Нека S_a е множеството от низове без буква a , S_b е множеството от низове без буква b и S_c е множеството от низове без буква c . Тогава $\overline{S_a}$ е множеството от низове с поне една буква a , $\overline{S_b}$ е множеството от низове с поне една буква b и $\overline{S_c}$ е множеството от низове с поне една буква c . Търсим

$$|\overline{S_a} \cap \overline{S_b} \cap \overline{S_c}|$$

Съгласно комбинаторния принцип на включването и изключването,

$$|\overline{S_a} \cap \overline{S_b} \cap \overline{S_c}| = |U| - (|S_a| + |S_b| + |S_c|) + (|S_a \cap S_b| + |S_a \cap S_c| + |S_b \cap S_c|) - |S_a \cap S_b \cap S_c|$$

Очевидно $|U| = 4^n$. Да намерим $|S_a|$. Но S_a е множеството от низове с дължина n над азбуката $\{b, c, d\}$, така че $|S_a| = 3^n$. Аналогично, $|S_b| = 3^n$ и $|S_c| = 3^n$.

Да намерим $|S_a \cap S_b|$. Но $S_a \cap S_b$ е множеството от низове с дължина n над азбуката $\{c, d\}$, така че $|S_a \cap S_b| = 2^n$. Аналогично, $|S_a \cap S_c| = 2^n$ и $|S_b \cap S_c| = 2^n$.

Да намерим $|S_a \cap S_b \cap S_c|$. Но $S_a \cap S_b \cap S_c$ е множеството от низове с дължина n над азбуката $\{d\}$. Има точно един такъв низ, а именно $dd \dots d$ с дължина n . Тогава $|S_a \cap S_b \cap S_c| = 1$.

Отговорът е

$$|\overline{S_a} \cap \overline{S_b} \cap \overline{S_c}| = 4^n - 3 \cdot 3^n + 3 \cdot 2^n - 1 = 4^n - 3^{n+1} + 3 \cdot 2^n - 1$$

Критерии за оценяване

Коректен обоснован отговор се оценява с 10 точки.

Коректен отговор без обосновка се оценява с 0 точки.

Некоректен отговор се оценява с 0 точки.

Задача 7. Даден е ориентиран граф, представен чрез поредица от факти от вида $\text{arc}(\langle \text{Node1} \rangle, \langle \text{Node2} \rangle, \langle \text{Cost} \rangle)$, всеки от които означава дъга с начало $\langle \text{Node1} \rangle$, край $\langle \text{Node2} \rangle$ и дължина (цена) $\langle \text{Cost} \rangle$:

$\text{arc}(s,a,1)$. $\text{arc}(s,d,3)$. $\text{arc}(s,f,4)$. $\text{arc}(a,b,2)$. $\text{arc}(d,e,2)$.
 $\text{arc}(f,g,4)$. $\text{arc}(b,c,1)$. $\text{arc}(e,g,1)$. $\text{arc}(c,g,1)$.

Дадена е също така поредица от факти от вида $h(\langle \text{Node} \rangle, \langle \text{Cost} \rangle)$, дефиниращи евристичната функция, с помощта на която се пресмята приближена стойност $\langle \text{Cost} \rangle$ на разстоянието от възела $\langle \text{Node} \rangle$ до възела g :

$h(s,4)$. $h(a,4)$. $h(b,2)$. $h(c,1)$. $h(d,3)$. $h(e,1)$. $h(f,4)$. $h(g,0)$.

Ако се търси път в графа от възела s до възела g , да се посочи решението, намерено с използване на всеки от посочените по-долу методи за търсене. Отговорът да се обоснове като за всеки един от алгоритмите се посочи как се избира следващият връх, който да се обходи, и каква е неговата оценяваща функция:

- a) best-first search;
- б) A^* search.

Примерно решение

Търсеният път е:

а) $[s, d, e, g]$

Методът Best-first search разширява този частичен път от фронта на търсене, който е с най-малка цена. Оценяващата функция е $f(n) = h(n)$, където $h(n)$ е евристика, оценяваща пътя от възела n до целта.

б) $[s, a, b, c, g]$

Методът A^* разширява този частичен път от фронта на търсене, който има най-малка стойност на оценяващата функция $f(n) = g(n) + h(n)$, където $g(n)$ е цената на пътя от началния връх до n , а $h(n)$ е евристика, оценяваща оставащата част от пътя до целта.

Критерии за оценяване

Всяка коректно отговорена и обоснована подточка се оценява с по 5 т.

Верни отговори без обосновка се оценяват с 2 т.

Задача 8. В пространството спрямо ортонормирана координатна система $K : O\vec{e}_1\vec{e}_2\vec{e}_3$ са дадени правите

$$m : \begin{cases} x = 3 + u \\ y = 2 + u \\ z = 2 \end{cases}, \quad g : \begin{cases} x + y + z - 2 = 0 \\ 3x - y - 11 = 0 \end{cases}.$$

Да се намери оста на правите m и g и дължината на тяхната ос-отсечка.

Примерно решение

Намираме параметрично представяне на правата g :

$$g : \begin{cases} x = v \\ y = -11 + 3v \\ z = 13 - 4v \end{cases}.$$

Произволна точка M от правата m има координати:

$$M(3 + u, 2 + u, 2).$$

Произволна точка G от правата g има координати:

$$G(v, -11 + 3v, 13 - 4v).$$

Следователно:

$$\overrightarrow{MG} = (-3 - u + v, -13 - u + 3v, 11 - 4v).$$

Понеже търсим ос на кръстосаните прави m и g трябва $\overrightarrow{MG} \perp m \parallel \vec{m}(1, 1, 0)$ и $\overrightarrow{MG} \perp g \parallel \vec{g}(1, 3, -4)$:

$$\begin{cases} \overrightarrow{MG} \vec{m} = (-3 - u + v, -13 - u + 3v, 11 - 4v)(1, 1, 0) = -16 - 2u + 4v = 0 \\ \overrightarrow{MG} \vec{g} = (-3 - u + v, -13 - u + 3v, 11 - 4v)(1, 3, -4) = -86 - 4u + 26v = 0 \end{cases}$$

Решението на тази система линейни уравнения е

$$u = -2, \quad v = 3$$

Намираме

$$M(1, 0, 2), \quad G(3, -2, 1)$$

За оста t намираме:

$$t : \begin{cases} \ni M(1, 0, 2), \\ \parallel \overrightarrow{MG}(2, -2, 1) \end{cases} \implies t : \begin{cases} x = 1 + 2s \\ y = -2s \\ z = 2 - s \end{cases}.$$

За дължината $|\overrightarrow{MG}|$ на оста-отсечка на кръстосаните прави m и g намираме

$$|\overrightarrow{MG}| = \sqrt{\overrightarrow{MG}^2} = \sqrt{2^2 + (-2)^2 + 1^2} = 3.$$

Критерии за оценяване

- За намиране параметрично представяне на правата $g \rightarrow 2$ т.
- За намиране на вектор $\overrightarrow{MG} \rightarrow 2$ т.
- За получаване на системата $\overrightarrow{MG} \perp \vec{m}$ и $\overrightarrow{MG} \perp \vec{g} \rightarrow 2$ т.
- За решаване на системата $\rightarrow 1$ т.
- За намиране на оста $\rightarrow 2$ т.
- За намиране на дължината на оста-отсечка $\rightarrow 1$ т.

Чернова