СОФИЙСКИ УНИВЕРСИТЕТ
“СВ. КЛИМЕНТ ОХРИДСКИ”

ФАКУЛТЕТ ПО МАТЕМАТИКА
И ИНФОРМАТИКА

ДЪРЖАВЕН ИЗПИТ

ЗА ПОЛУЧАВАНЕ НА ОКС “БАКАЛАВЪР ПО ИНФОРМАЦИОННИ СИСТЕМИ”

ЧАСТ I (ПРАКТИЧЕСКИ ЗАДАЧИ)

Драги абсолвенти:

- Попълнете факултетния си номер в горния десен ъгъл на всички листове.
- Пишете само на предоставените листове, без да ги разкопчавате.
- Решението на една задача трябва да бъде на същия лист, на който е и нейното условие (т.е. може да пишете отпред и отзад на листа със задачата, но не и на лист на друга задача).
- Ако имате нужда от допълнителен лист, можете да поискате от квесторите.
- На един лист не може да има едновременно и чернова, и белова.
- Черновите трябва да се маркират, като най-отгоре на листа напишете “ЧЕРНОВА”.
- Ако решението на една задача не се побира на нейния лист, трябва да поискате нов бял лист от квесторите. Той трябва да се 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++. 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.

1) Дадена е дефиницията:

```
void print(int* arr, int size) {
  for (int i = 0; i < size; ++i)
    cout << arr[i] << ";";
  cout << "\n";
}
```

Какво ще изведе на стандартния изход даденият по-долу фрагмент? Всеки отделен ред от изхода да се попълни в съответното поле.

```
const int size = 5;
int arr[] = { 5, 10, -1, 5, 6 };
for (int i = 0; i < size; ++i) {
  for (int j = 1; j < size; ++j) {
    if (arr[j-1] < arr[j]) {
      swap(arr[j-1], arr[j]);
    }
  }
  print(arr, size);
}
```

На ред 1 извежда: _____
На ред 2 извежда: _____
На ред 3 извежда: _____
На ред 4 извежда: _____
На ред 5 извежда: _____

2) Какво трябва да се попълни на мястото на коментара в `toUpper` така, че ако на функцията се подаде малка латинска буква, тя да връща съответната ѝ главна, а всички други символи да връща непроменени?

```
char toUpper(char c) {
  return /* какво трябва да има тук? */ ;
}
```

А) `(c >= 'a' || c <= 'z') ? (c - 'a' + 'A') : c`
Б) `(c >= 'a' || c <= 'z') ? (c + 'A') : c`
В) `(c >= 'a' || c <= 'z') ? (c - 'a') : c`
Г) `(c >= 'a' && c <= 'z') ? (c - 'a' + 'A') : c`[1]
Д) `(c >= 'a' && c <= 'z') ? (c + 'A') : c`[1]
Е) `(c >= 'a' && c <= 'z') ? (c - 'a') : c`[1]

3) Да се попълнят празните места в дефинициите на функциите така, че `isOdd` да връща истина тогава и само тогава, когато подаденото ѝ число е нечетно, а `isEven` — когато то е четно.

```
bool isEven(unsigned int);
bool isOdd(unsigned int n) {
  if (n == 0) return ________;
  if (n == 1) return ________;
  return isEven(________);
}
bool isEven(unsigned int n) {
  if (n == 0) return ________;
  if (n == 1) return ________;
  return isOdd(_________);
}
```

4) Под всеки от дадените по-долу фрагменти да се посочи какво ще изведе той на стандартния изход.

```
for (int i = 0; i < 9; ++i)
  cout << (1 << i) << ";";
```

_____

```
switch(5 % 2) {
case 0:
  std::cout << "0";
case 1:
  std::cout << "1";
default:
  std::cout << "x";
}
```

_____

```
for (int i = 0; i < 5; ++i) {
  if ( ! (i - 3))
    continue;
  std::cout << i;
}
```

_____

```
int arr[] = { 1, 2, 3, 4, 0 };
*(arr + 2) *= 10;
for (int* p = arr; *p; p++)
  std::cout << *p << ";";
```

_____

[1] В дадената на изпита тема на тези отговори беше допусната печатна грешка, която тук е отстранена.

Критерии за оценяване

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. Ако изходът от реда не съвпада с това, което е посочено в отговора, той се оценява с 0 т. Сумата от точките се закръгля до цяло число.

По-конкретно за съответните подточки:

1) Всеки напълно коректно посочен ред носи 0,4 т. (общо 2 т.). Изходът е както следва:

а) На ред 1 извежда: `10;5;5;6;-1;`
б) На ред 2 извежда: `10;5;6;5;-1;`
в) На ред 3 извежда: `10;6;5;5;-1;`
г) На ред 4 извежда: `10;6;5;5;-1;`
д) На ред 5 извежда: `10;6;5;5;-1;`

2) Носи 1 т. ако е посочен коректният отговор и 0 т. в противен случай. Коректният отговор е:

```
(c >= 'a' && c <= 'z') ? (c-'a'+'A') : c
```

3) Всяко напълно коректно попълнено място носи 0,5 т. (общо 3 т.). Ако в някой отговор вместо булевите литерали `true` и `false` са използвани 0 или 1, точките за отговора се намаляват наполовина. Примерно решение:

```
bool isEven(unsigned int);
bool isOdd(unsigned int n) {
  if (n == 0) return false;
  if (n == 1) return true;
  return isEven(n - 1);
}
bool isEven(unsigned int n) {
  if (n == 0) return true;
  if (n == 1) return false;
  return isOdd(n - 1);
}
```

4) Всеки напълно коректно отговорен фрагмент носи 1 т. (общо 4 т.). Изходът от фрагментите е посочен по-долу:

```
for (int i = 0; i < 9; ++i)
  cout << (1 << i) << ";";
```

```
1;2;4;8;16;32;64;128;256;
```

```
switch(5 % 2) {
case 0:
  std::cout << "0";
case 1:
  std::cout << "1";
default:
  std::cout << "x";
}
```

```
1x
```

```
for (int i = 0; i < 5; ++i) {
  if ( ! (i - 3))
    continue;
  std::cout << i;
}
```

```
0124
```

```
int arr[] = { 1, 2, 3, 4, 0 };
*(arr + 2) *= 10;
for (int* p = arr; *p; p++)
  std::cout << *p << ";";
```

```
1;2;30;4;
```

Задача 2.

Задачата да се реши на езика C++. Дадени са дефинициите:

```
class A {
public:
  A()   { cout << "A()\n"; }
  A(A&) { cout << "A(A&)\n"; }
  virtual ~A() { cout << "~A()\n"; }
  A& operator=(A&) {
    cout << "op=(A&)\n";
    return *this;
  }
};

class B : public A {
public:
  B()   { cout << "B()\n"; }
  B(B&) { cout << "B(B&)\n"; }
  virtual ~B() { cout << "~B()\n"; }
  B& operator=(B&) {
    cout << "op=(B&)\n";
    return *this;
  }
};

void f(A b) { cout << "f(A)\n"; }
```

Под всеки от редовете на дадения вдясно програмен фрагмент да се посочи какво ще се изведе в резултат от неговото изпълнение. (Между редовете нарочно е оставено повече място, за да може да попълните отговора си) Ако смятате, че някой ред няма да изведе нищо, напишете "не извежда нищо". Ако смятате, че някой от редовете ще предизвика грешка, напишете "грешка" и обяснете каква е тя и защо възниква.

За коректни се считат отговорите, които напълно съответстват на това, което ще се случи за съответния ред. Текст "грешка" без обяснение носи нула точки.

```
B d;
```

```
B copy = d;
```

```
A b = d;
```

```
A& ref = d;
```

```
B arr[2];
```

```
f(d);
```

```
A* p = new B(d);
```

```
delete p;
```

```
d = d;
```

```
ref = d;
```

Критерии за оценяване

- Всеки ред, за който напълно коректно е посочено какво ще се изведе/случи, носи 1 т.
- Ако изходът от реда не съвпада с това, което е посочено в отговора, той се оценява с 0 т.
- Текст “Грешка” без обяснение към него се оценява с 0 т.

Изходът от редовете е посочен по-долу:

```
Изход от "B d":
  A()
  B()

Изход от "B copy = d":
  A()
  B(B&)

Изход от "A b = d":
  A(A&)

Изход от "A& ref = d":
  не извежда нищо

Изход от "B arr[2]":
  A()
  B()
  A()
  B()

Изход от "f(d)":
  A(A&)
  f(A)
  ~A()

Изход от "A* p = new B(d)":
  A()
  B(B&)

Изход от "delete p":
  ~B()
  ~A()

Изход от "d = d":
  op=(B&)

Изход от "ref = d":
  op=(A&)
```

Задача 3. Извършват се последователни опити. При всеки опит се хвърлят четири правилни монети. Считаме, че опитът е успешен, ако при него се паднат равен брой "лица" и "гербове".

а) Да се определи вероятността един опит да е успешен.

б) Каква е вероятността да са се случили точно три неуспешни опита преди втори успешен опит?

в) Нека X е броят на успешните опити до първия неуспешен опит. Да се пресметнат математическото очакване $\mathrm{E}X$ и дисперсията $\mathrm{D}X$.

Примерно решение:

а) (3 точки) Хвърлят се четири монети. Вероятността за падане на герб на всяка от тях е $\frac{1}{2}$. Броя паднали се "гербове" да съвпадне с броя на падналите се "лица", означава да се паднат точно два герба и две лица. От четирите монети избираме двете върху който да са гербовете по $\binom{4}{2}$ начина. Вероятността да се падне герб на две фиксирани монети е $\left(\frac{1}{2}\right)^2$. Аналогично, вероятността за лице на останалите две монети е $\left(\frac{1}{2}\right)^2$. Окончателно за вероятността за успех p получаваме:

$$p = \binom{4}{2}\left(\frac{1}{2}\right)^2\left(\frac{1}{2}\right)^2 = \frac{3}{8}$$

За намиране на тази вероятност може да се използва и биномно разпределение. Да означим с Z броя на падналите се гербове при хвърляне на четири монети. Не е трудно да се разпознае, че случайната величина Z е биномно разпределена $Z \in Bi(4, 1/2)$ и търсената вероятност е $p = P(Z = 2)$.

б) (3 точки) Съгласно предишната подточка вероятността за успех е $p = \frac{3}{8}$, съответно за неуспех $q = \frac{5}{8}$. За да има точно три неуспеха преди втория успех явно са проведени пет опита, при което на последния опит има успех. Тогава трябва да намерим вероятността на събитието — при първите четири опита има точно един успех и последния опит е успех. За първите четири опита използваме отново биномна вероятност, а при последния опит вероятността за успех е $\frac{3}{8}$. Така получаваме:

$$\binom{4}{1}pq^3\,p = \binom{4}{1}\left(\frac{3}{8}\right)^2\left(\frac{5}{8}\right)^3 = \frac{1125}{8192}$$

в) (4 точки) Случайната величина X отговаря на модела на геометричното разпределение, единствената разлика е, че в стандартния случай се търси броя на неуспехите до първия успех, а в тази задача търсим броя на успехите до първия неуспех. Тогава $X \in Ge\left(\frac{5}{8}\right)$. За намиране на очакването и дисперсията се използват директно (не е нужно да се извеждат) формулите, известни от свойствата на геометрично разпределение. При означенията въведени по-горе:

$$\mathrm{E}X = \frac{p}{q} = \frac{3/8}{5/8} = \frac{3}{5},$$

$$\mathrm{D}X = \frac{p}{q^2} = \frac{3/8}{(5/8)^2} = \frac{24}{25}.$$

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

Група приятели отиват на екскурзия и се редуват да плащат общите сметки. Когато се прибират, установяват, че всеки дължи на някого дребна сума пари. Всяко задължение се представя с кортеж от три елемента: длъжник (низ), сума (цяло положително число) и получател (низ). За Scheme под "кортеж" се има предвид просто списък. "Верига" наричаме последователност от **различни** хора, в която всеки дължи на предишния една и съща сума. "Кръг" наричаме затворена верига, т.е. верига, в която първият дължи на последния същата сума. Приятелите се сещат, че ако в задълженията им има верига, то може последният човек в нея да се издължи директно на първия, а ако има кръг, то всички задължения в него се погасяват. Да се попълнят празните полета по-долу така, че:

а) функцията `maxByLength` да намира най-дълъг списък в списъка от списъци `ls`;
б) рекурсивната функция `maxChain` да намира най-дълга верига, завършваща със списъка `chain`, сума `amount` и краен получател (т.е. първи елемент) `final` при даден списък от задължения `dues`;
в) функцията `maxCircle` да намира най-дълъг кръг при даден списък от задължения `dues`.

За всички функции, ако има няколко различни отговора с еднаква дължина, не е от значение кой от тях ще бъде върнат като резултат, а ако няма нито един отговор, да се връща празният списък.

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

Haskell

```
maxByLength ls = foldr ____________________________________________________ [] ls
maxChain chain@(last:rest) amount final dues
 | ____________________________ = chain
 | otherwise   = maxByLength [ maxChain ____________________________ amount final dues  |
                              ____________________________ <- dues, amount == _________,
                              last == _________, ____________________________ ]
maxCircle dues = maxByLength [ maxChain _________ _________ _________ dues |
                               ____________________________ <- dues ]
```

Пример:

```
dues = [("Georgi",10,"Boyan"),("Boyan",15,"Sia"),("Sia",15,"Maria"),("Maria",10,"Georgi"),
        ("Maria",10,"Petar"),("Petar",10,"Georgi"),("Boyan",10,"Maria")]
maxChain ["Boyan"] 15 "Maria" dues ⟶ ["Maria","Sia","Boyan"]
maxCircle dues ⟶ ["Boyan","Georgi","Petar","Maria"]
```

Scheme

```
(define (maxByLength ls)
  (foldr ____________________________________________________ '() ls))
(define (maxChain chain amount final dues)
  (let ((last (car chain)) (rest (cdr chain)))
    (if ____________________________ chain
        (maxByLength (map (lambda (t) (maxChain ____________________________ amount final dues))
                          (filter (lambda (t) (and (= amount _________)
                                                   (eqv? last _________)
                                                   ____________________________)) dues))))))
(define (maxCircle dues)
  (maxByLength (map (lambda (t)
                      (maxChain _________ _________ _________ dues) dues))))
```

Пример:

```
(define dues '(("Georgi" 10 "Boyan") ("Boyan" 15 "Sia") ("Sia" 15 "Maria") ("Maria" 10 "Georgi")
               ("Maria" 10 "Petar") ("Petar" 10 "Georgi") ("Boyan" 10 "Maria")))
(maxChain '("Boyan") 15 "Maria" dues) ⟶ ("Maria" "Sia" "Boyan")
(maxCircle dues) ⟶ ("Boyan" "Georgi" "Petar" "Maria")
```

Примерни решения

Haskell

```
maxByLength ls = foldr (\x r -> if length x > length r then x else r) [] ls

maxChain chain@(last:rest) amount final dues
 | last == final = chain
 | otherwise     = maxByLength [ maxChain (receiver:chain) amount final dues  |
                                 (giver, due, receiver) <- dues, amount == due,
                                 giver == last, not (receiver `elem` rest) ]

maxCircle dues = maxByLength [ maxChain [receiver] due giver dues |
                               (giver, due, receiver) <- dues ]
```

Scheme

```
(define (maxByLength ls)
  (foldr (lambda (x r) (if (> (length x) (length r)) x r)) '() ls))

(define (maxChain chain amount final dues)
  (let ((last (car chain)) (rest (cdr chain)))
    (if (eqv? last final) chain
        (maxByLength (map (lambda (t)
                            (maxChain (cons (caddr t) chain) amount final dues))
                          (filter (lambda (t) (and (= amount (cadr t))
                                                   (eqv? last (car t))
                                                   (not (memv (caddr t) rest)))) dues))))))

(define (maxCircle dues)
  (maxByLength (map (lambda (t)
                      (maxChain (list (caddr t)) (cadr t) (car t) dues)) dues)))
```

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:

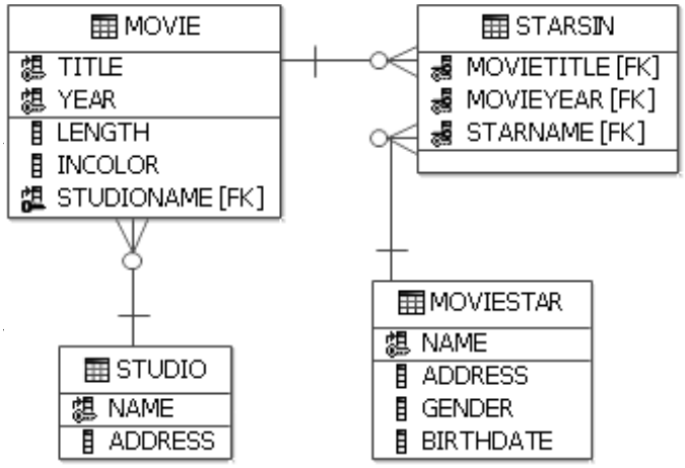
1. **3 т.** за функцията `\x r -> if length x > length r then x else r` или еквивалентна на нея, която връща този от параметрите, който е с по-голяма (или не по-малка) дължина;
2. **1 т.** за сравнението `last == final` или симетричното му;
3. **1 т.** за конструкцията `(receiver:chain)` или еквивалентна, където `receiver` е третият елемент от слот 4, не се дават точки при синтактична грешка в слот 4;
4. не се дават или отнемат точки, служи само за именуване на променливите, които се използват в другите слотове;
5. **0,5 т.** за попълнен вторият елемент от слот 4, не се дават точки при синтактична грешка в слот 4;
6. **0,5 т.** за попълнен първият елемент от слот 4, не се дават точки при синтактична грешка в слот 4;
7. **2 т.** за проверката `not (receiver `elem` rest)` или еквивалентна проверка за избягване на зациклянето, където `receiver` е третият елемент от слот 4, не се дават точки при синтактична грешка в слот 4;
8. **1 т.** за конструкцията `[receiver]` или еквивалентна, където `receiver` е третият елемент от слот 11, не се дават точки при синтактична грешка в слот 11;
9. **0,5 т.** за попълнен вторият елемент от слот 11, не се дават точки при синтактична грешка в слот 11;
10. **0,5 т.** за попълнен първият елемент от слот 11, не се дават точки при синтактична грешка в слот 11;
11. не се дават или отнемат точки, служи само за именуване на променливите, които се използват в другите слотове.

Scheme

По ред на празните слотове:

1. **3 т.** за функцията `(lambda (x r) (if (> (length x) (length r)) x r))` или еквивалентна на нея, която връща този от параметрите, който е с по-голяма (или не по-малка) дължина;
2. **1 т.** за сравнението `(eqv? last final)` или симетричното му, позволено е използването и на `equal?`, **0,5 т.** при използване на `=` или `eq?`;
3. **1 т.** за конструкцията `(cons (caddr t) chain)` или еквивалентна;
4. **0,5 т.** за конструкцията `(cadr t)` или еквивалентна;
5. **0,5 т.** за конструкцията `(car t)` или еквивалентна;
6. **2 т.** за проверката `(not (memv (caddr t) rest)` или еквивалентна проверка за избягване на зациклянето, позволено е използването и на `member`, **1 т.** при използване на `memq`;
7. **1 т.** за конструкцията `(list (caddr t))` или еквивалентна;
8. **0,5 т.** за конструкцията `(cadr t)` или еквивалентна;
9. **0,5 т.** за конструкцията `(car t)` или еквивалентна.

Задача 5. Дадена е базата от данни `Movies`, в която се съхранява информация за филми, филмови студиа, които ги произвеждат, както и актьорите, които участват в тях.



Таблицата `Studio` съдържа информация за филмови студиа:

- `name` — име, първичен ключ
- `address` — адрес;

Таблицата `Movie` съдържа информация за филми. Атрибутите `title` и `year` заедно формират първичния ключ.

- `title` — заглавие
- `year` — година, в която е заснет филмът
- `length` — дължина в минути
- `incolor` — `'Y'` за цветен филм и `'N'` за чернобял
- `studioname` — име на студио, външен ключ към `Studio.name`;

Таблицата `MovieStar` съдържа информация за филмови звезди:

- `name` — име, първичен ключ
- `address` — адрес
- `gender` — пол, `'M'` за мъж (актьор) и `'F'` за жена (актриса)
- `birthdate` — рождена дата.

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

- `movietitle` — заглавие на филма
- `movieyear` — година на заснемане на филма
- `starname` — име на филмовата звезда, външен ключ към `MovieStar.name`.

1) Да се попълнят празните места в следната заявка така, че тя да извежда името на студиото на филма 'The Usual Suspects' и заглавията на всички филми на същото студио:

```
SELECT s.name, m.title
FROM movie ____ JOIN studio s ON m.studioname ____________________
WHERE s.name ____ (SELECT _________________
                   FROM   _________________
                   WHERE title = 'The Usual Suspects' AND year = 1995);
```

2) Да се посочи коя от следните заявки извежда имената на филмовите звезди, за които няма информация в кои филми са играли:

```
А) SELECT DISTINCT starname
   FROM starsin
   GROUP BY starname
   HAVING COUNT(*) = 0;
```

```
Б) SELECT ms.name, si.movietitle
    FROM moviestar ms
    LEFT JOIN starsin si
           ON ms.name=si.starname
    WHERE si.movietitle IS NULL;
```

```
В) SELECT name
   FROM starsin
   JOIN moviestar ON starname = name
   GROUP BY name
   HAVING COUNT(name) = 0;
```

```
Г) SELECT name
   FROM moviestar
   WHERE NOT EXISTS (SELECT starname
                     FROM starsin);
```

Критерии за оценяване

а) Общо 5 т., по ред на празните слотове:
- `m` или `AS m` — **1 т.**
- `=s.name` или `= name` — **1 т.**
- `=` или `IN` — **1 т.**
- `studioname` — **1 т.**
- `movie` — **1 т.**

б) Единственият верен отговор е Б). Посочването на този отговор носи 5 т., а посочването на грешен отговор или комбинация от отговори носи 0 т.

Заявката, описана в този отговор, коректно извежда името на филмовите звезди, за които няма информация в кои филми са играли, съгласно условието на тази подточка. Фактът, че в допълнение на исканата в условието информация заявката извежда също и втора колона с име `movietitle` и стойности `NULL`, не променя коректността на заявката относно зададеното така условие.

Заявките, посочени във всички останали отговори, не отговарят на условието на подточката. В частност, заявките в отговор А) и отговор В) винаги връщат празно множество, независимо дали има или не филмови звезди, за които няма информация в кои филми са играли, както би трябвало да е по условие. Заявката в отговор Г) не отговаря на условието, понеже в случая, в който таблицата `starsin` не е празна, не се извежда името на нито една филмова звезда, вместо да се изведат имената на тези, за които няма информация в кои филми са играли, както би трябвало да е по условие. Това поведение е съгласно стандарта за езика SQL и не зависи от конкретната реализация в система за управление на бази от данни.

Задача 6. Информационна система съхранява информация за сайт за резервации на хотелски стаи от клиенти. Съхранява се информация за клиентите на сайта: email на клиент – низ до 30 символа, уникален за всеки клиент, име на клиент – низ до 50 символа. Съхранява се информация и за хотелите, регистрирани в сайта: име на хотела – низ до 30 символа, уникално за всеки хотел, адрес на хотела – низ до 50 символа и брой звезди – цяло положително число не повече от 7. Всеки хотел разполага с различни по тип стаи, които могат да бъдат резервирани от клиенти за даден период от време. Всяка стая се характеризира с номер на стая – цяло положително число, тип на стаята – низ до 10 символа и цена на стаята – реално положително число. Номерът на стаята е уникален в рамките на хотела, в който се намира съответната стая. Стаите могат да бъдат с едно легло, с две легла, студио, апартамент – това е отразено в типа на стаята. Клиентите резервират хотелски стаи през сайта. За резервацията се пази информация за номер на резервацията – низ точно 8 символа, уникален за всяка резервация, дата на резервацията, дата, указваща откога е резервирана стаята, дата, указваща докога е резервирана стаята, статус на резервацията – низ точно 5 символа, дължима сума за броя дни престой – реално положително число.

Задание:

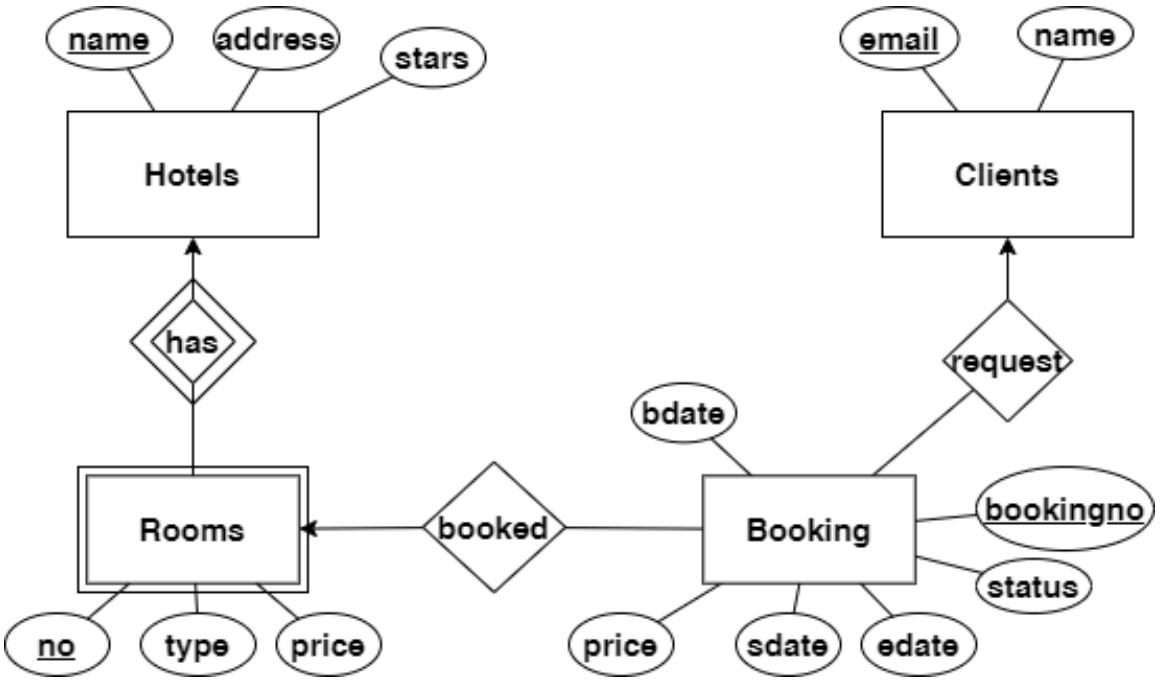
а) Да се направи E/R модел на БД, която съхранява гореописаната информация. Да се начертае E/R диаграма на модела.

б) Да се преобразува E/R диаграмата към релационни схеми. Да се премахнат излишествата, където това е възможно.

в) Да се напише DDL код, съответстващ на релационните схеми. Да се реализират всички описани ограничения.

Примерно решение:

а)



б) Hotels (<u>name</u>, address, stars)
Rooms (<u>nameHotel</u>, <u>no</u>, type, price)
Clients (<u>email</u>, name)
Booking (<u>bookingno</u>, bdate, sdate, edate, price, status, clientEmail, nameHotel, roomNo)

в)
```
CREATE TABLE HOTELS (
    name VARCHAR(30) NOT NULL PRIMARY KEY,
    address VARCHAR(50) NOT NULL,
    stars INT CHECK(STARS >= 0 AND STARS < 8)
);

CREATE TABLE ROOMS (
    nameHotel VARCHAR(30) NOT NULL REFERENCES HOTELS(NAME),
    no INT CHECK (NO > 0) NOT NULL,
    type VARCHAR(10),
    price DECIMAL(9,2) CHECK(PRICE > 0),
    PRIMARY KEY(NAMEHOTEL, NO)
);

CREATE TABLE CLIENTS (
    email VARCHAR(30) NOT NULL PRIMARY KEY,
    name VARCHAR(50) NOT NULL
);

CREATE TABLE BOOKING (
    bookingno CHAR(8) NOT NULL PRIMARY KEY,
    bdate DATE  NOT NULL,
    sdate DATE NOT NULL,
    edate DATE NOT NULL,
    price DECIMAL(9,2) CHECK(PRICE > 0),
```

```
    status CHAR(5) NOT NULL,
    clientEmail VARCHAR(30) NOT NULL REFERENCES CLIENTS(EMAIL),
    nameHotel VARCHAR(30) NOT NULL,
    roomNo INT NOT NULL,
    FOREIGN KEY(NAMEHOTEL, ROOMNO) REFERENCES ROOMS(NAMEHOTEL, NO)
);
```

Критерии за оценяване:

а) 4 т. от които:

- до 2 т. за коректно отразяване на информацията в условието на задачата, като
  - 1 т. се дават за частично отразяване на информацията
  - 0 т. се дават за основни грешки в отразяване на информацията
- до 2 т. за коректно реализирана диаграма, като
  - 1 т. се дават за пропуски в E/R диаграмата
  - 0 т. се дават за съществени пропуски в E/R диаграмата

б) 3 т. от които:

- 1 т. за коректно преобразуване на множествата от същности към релации
- +1 т. за коректно представени и оптимизирани връзки “едно към много”
- +1 т. за коректно преобразувано слабо множество към релация

в) 3 т. от които:

- 1 т. за коректно описание на колоните, типа на колоните и първичните ключове
- +1 т. за коректно описание на външните ключове
- +1 т. за коректно описание на check ограниченията

Задача 7. Разглеждаме Информационната система за резервации, представена в Задача 6. Системата дава възможност:

- да се регистрира място за настаняване;
- да се предложи дадено място за настаняване за резервации за определен период или за постоянно;
- да се управляват резервациите, направени от клиент;
- клиентът да разгледа предлаганите места за настаняване, след като зададе желани критерии за търсене (например локация, дати на наемане, тип, брой гости, удобства, правила за анулиране);
- клиентът да резервира стая в избрано място за настаняване;
- клиентът да анулира резервацията преди датата на настаняване.

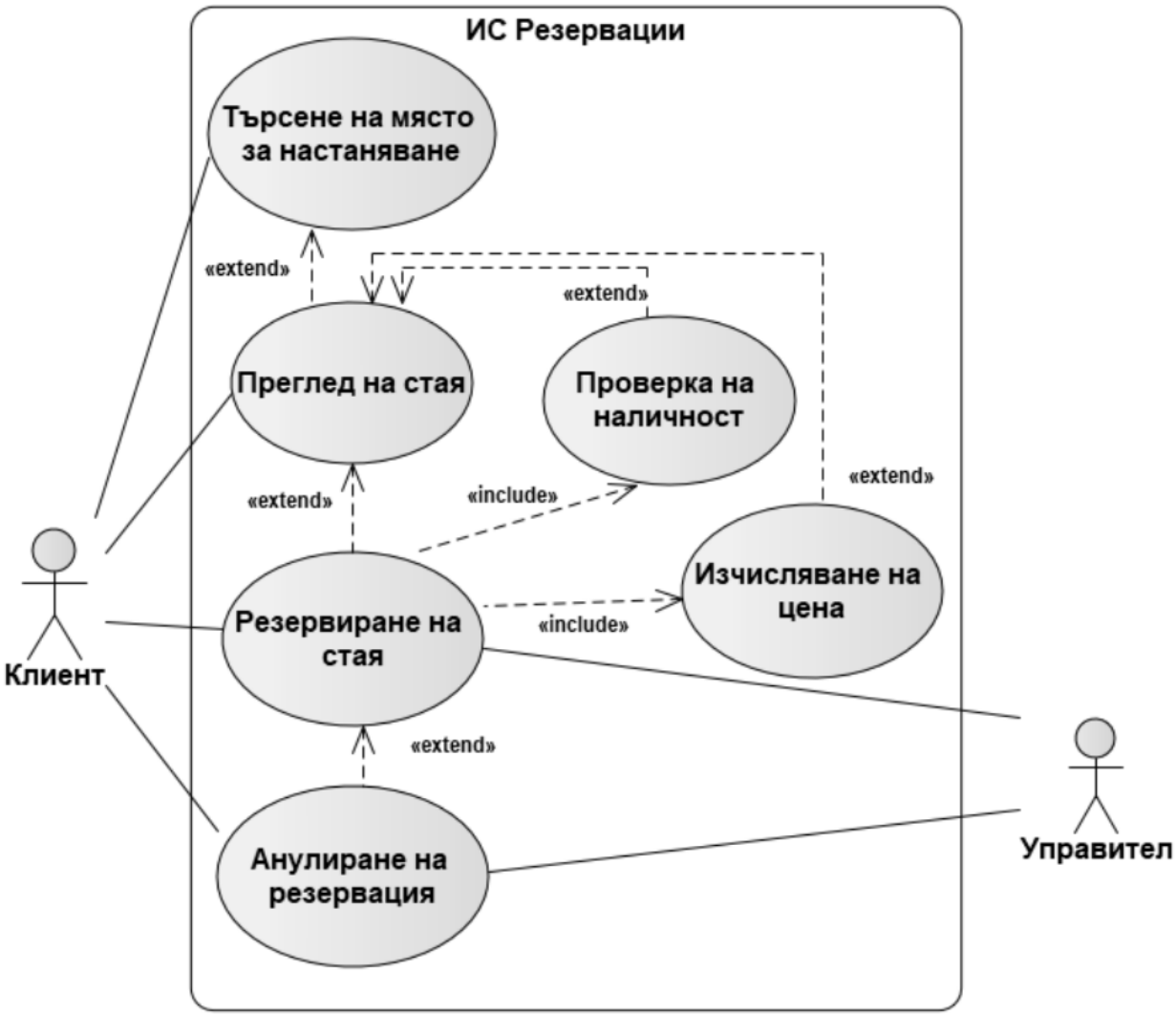
За извършена, както и за анулирана резервация се уведомяват клиентът и управителят на мястото за настаняване.

а) Да се подготви диаграма на потребителските случаи, които се асоциират с клиента. На диаграмата да се представят: актьори; потребителски случаи; отношения между потребителските случаи. Диаграмата да се подготви в термините на UML стандарта.

б) Да се опише в пълен формат потребителски случай за резервиране на стая от клиент. В описанието да се включат: актьори; предусловия; следусловия (резултати); основен сценарий; отношения с други потребителски случаи (ако е приложимо); алтернативен сценарий.

Примерно решение:

а)



б)

Актьори	Клиент, Управител
Предусловия	UC Преглед на стая
Следусловия	Обектът "Стая" е в състояние "Заета" за избрания период на наемане
Основен сценарий	
1	Клиентът избира опция "Резервирай"
2	Система отваря форма за резервация
3	Клиентът попълва форма за резервация
4	includes UC "Проверка на наличност" Системата връща съобщение за наличност
5	includes UC "Изчисляване на цена"
6	Системата извежда за преглед цялата информация за резервацията
7	Клиентът избира опция "Завърши"
8	Системата генерира номер на резервация
9	Системата извежда уведомително съобщение за успешна резервация
10	Системата изпраща до клиент и управител уведомително съобщение за направената резервация
11	Системата променя състоянието на стаята на "Заета" за избрания период на наемане
Алтернативен сценарий	
4.1	Системата връща съобщение за неналичност
4.2	Изпълнението се връща в т.3 от основния сценарий

Критерии за оценяване

а) точките се изчисляват по следните критерии и се закръглят до цяло число:

- съответствие с описания сценарий: 0,25 т.
- съответствие с условието на задачата: 0,25 т.
- съответствие с UML нотацията: 0,50 т.
- съответствие с изискванията за подготовка на модел на потребителските случаи (ПС) в частта диаграма на ПС, включително:
  - коректно определяне на границите на системата: 0,75 т.
  - коректно определяне на актьорите: 0,50 т.
  - коректна разбивка на функционалността на системата на ПС: 2,00 т.
  - коректно определяне на отношенията между ПС: 0,75 т.

б) точките се изчисляват по следните критерии и се закръглят до цяло число:

- съответствие с описания сценарий: 0,25 т.
- съответствие с условието на задачата: 0,25 т.
- съответствие с изискванията за подготовка на модел на потребителските случаи (ПС) в частта пълно описание на ПС, включително:
  - коректно са определени актьори: 0,25т.
  - коректно са определени предусловия и следусловия: 0,25 т.
  - коректно е определена функционалността на системата, която обхваща отделният потребителски случай: 2,50 т.

– основен и алтернативен сценарии са описани в отделни стъпки и всеки от тях представлява напълно завършен сценарий: 0,25 т.
– отношенията с други ПС са посочени в предусловие или на стъпките от сценария, където се извършва разклонение или извикване: 1,00 т.
– съответствие с диаграмата на ПС от подусловие а): 0,25 т.

Задача 8. Да се пресметне определеният интеграл

$$\int_0^1 \sqrt{x}\,\ln\sqrt{1+x}\,dx.$$

Примерно решение

В интеграла правим субституцията $x = t^2$, $t \geq 0$ Използваме, че $\ln \sqrt{1+x} = \frac{1}{2} \ln(1+x)$. Така получаваме

$$\int_0^1 \sqrt{x} \ln \sqrt{1+x}\, dx = \int_0^1 t^2 \ln(1+t^2)\, dt.$$

След това внасяме t^2 под знака на диференциала и интегрираме по части:

$$\begin{aligned}\int_0^1 t^2 \ln(1+t^2)\, dt &= \frac{1}{3} \int_0^1 \ln(1+t^2)\, d(t^3) \\ &= \frac{1}{3} \left(t^3 \ln(1+t^2) \Big|_0^1 - \int_0^1 t^3 d \ln(1+t^2) \right) \\ &= \frac{\ln 2}{3} - \frac{2}{3} \int_0^1 \frac{t^4}{1+t^2}\, dt.\end{aligned}$$

Използваме разлагането

$$\frac{t^4}{1+t^2} = \frac{t^4-1}{t^2+1} + \frac{1}{1+t^2} = t^2 - 1 + \frac{1}{1+t^2}.$$

Така получаваме

$$\begin{aligned}\int_0^1 t^2 \ln(1+t^2)\, dt &= \frac{\ln 2}{3} - \frac{2}{3} \int_0^1 t^2\, dt + \frac{2}{3} \int_0^1 dt - \frac{2}{3} \int_0^1 \frac{dt}{1+t^2} \\ &= \frac{\ln 2}{3} - \frac{2}{3} \frac{t^3}{3} \Big|_0^1 + \frac{2}{3} - \frac{2}{3} \operatorname{arctg} t \Big|_0^1 \\ &= \frac{\ln 2}{3} - \frac{2}{9} + \frac{2}{3} - \frac{2}{3} \left(\frac{\pi}{4} - 0 \right) \\ &= \frac{\ln 2}{3} + \frac{4}{9} - \frac{\pi}{6}.\end{aligned}$$

Критерии за оценяване

- субституция $x = t^2$: 2 т.,
- интегриране по части: 3 т.,
- разлагане на $\frac{t^4}{1+t^2}$ във вид удобен за интегриране (в сума на елементарни дроби): 2 т.
- пресмятане на $\int_0^1 (t^2 - 1)\, dt$: 1 т.,
- пресмятане на $\int_0^1 \frac{1}{1+t^2}\, dt$: 1 т.,
- получаване на окончателния отговор: 1 т.

Чернова