Държавен изпит на ОКС бакалавър по Информационни системи (ИС), 16-17.07.2013

Държавен изпит на ОКС бакалавър по Информационни системи (ИС), 16-17.07.2013

Fold

Table of Contents
Съдържание

Задачите са възстановени по спомен. Всяка задача се оценява по шестобална система (2-6), а общата оценка от задачи се формира на база тежест (в проценти) на всяка задача. Логично е сумата на процентите да е 100, но тук са възстановявани по спомен и излизат към 97. Времето за работа е 3 астрономически часа.

Задачи

Зад. 1 (16%, C/C++)

Даден е код на функция на C, която сравнява 2 низа (или символи в низ, нещо такова...). Да се напише функция на C или C++, която по зададен масив от низове (char*), дължина на масива (цяло число) и указател към функция, задаваща наредба (comparator), сортира масива.

Зад. 2 (16%, C++)

Да се създаде клас Task, който представя за изпълнение. Съхранява информация за име на задачата (низ до 100 символа) и време за изпълнение на задачата (цяло число). Да се създаде и друг клас, RecurringTask, представящ задача, която се повтаря. В допълнение, класът съхранява информация за броя повторения на задачата (цяло число). Да се дефинират:

- подходящи конструктори;
- метод `print`, който печата информация задача;
- метод `getTotalTime`, който връща общото време за изпълнение на задача (за RecurringTask е времето, умножено по броя пъти);
- метод `printShort`, който за масив от задачи отпечатва информация за тези от тях, чието общо време за изпълнение е по-малко от общото време на обекта, за който се вика методът. Масивът може да съдържа обекти както от Task, така и от RecurringTask.

Зад. 3 (17%, C/C++)

Двоично дърво е представено по следния начин:

```
struct Node
{
    int data; // или пък char* да е било, принципно е без значение
    Node* left;
    Node* right;
    Node* parent;
}
```

`left` е указател към ляв възел (поддърво), `right` - към десен, `parent` - към родител. Да се напише функция

```
Node* lowestCommonAncestor ( Node* n1, Node* n2 )
```

, която връща най-близкия общ предшественик на два възела (`n1` и `n2`). За реализацията може да се използват структури от STD наготово (бяха изброили кои точно).

Зад. 4 (7 или 8%, Haskell)

За списък от цели числа [x_1,x_2,...,x_n] едноместна функция `f` казваме, че "запазва" елемент, ако `f(x)` също принадлежи на списъка. Да се напишат следните функции на Haskell:

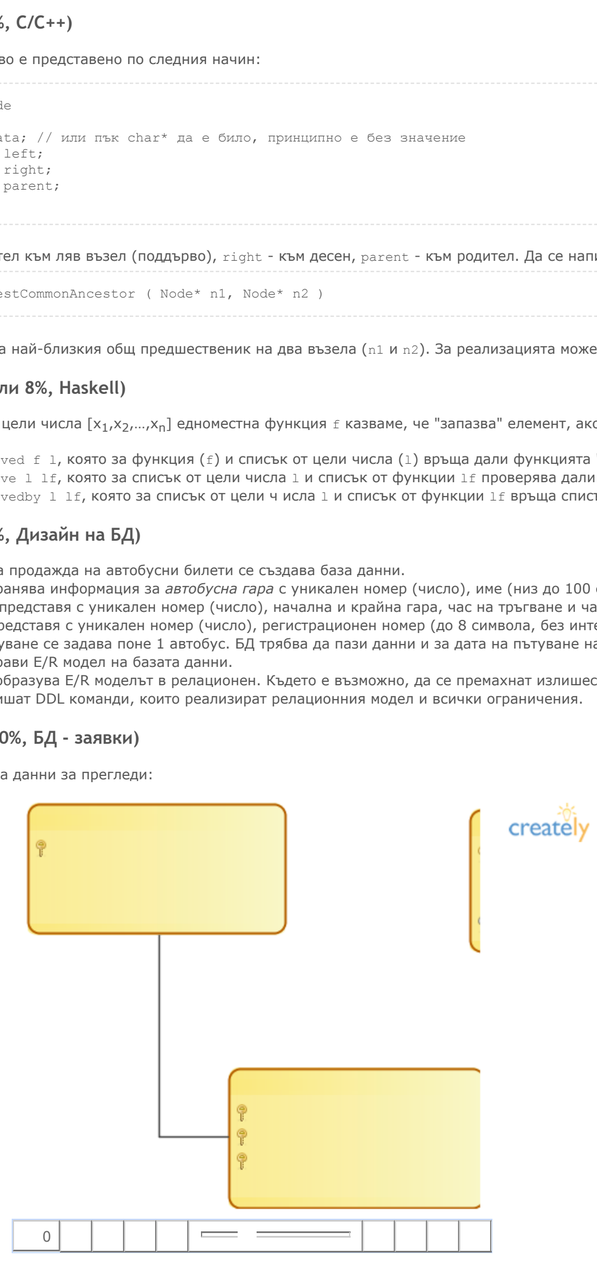
- `preserved f l`, която за функция (`f`) и списък от цели числа (`l`) връща дали функцията "запазва" елементите на l;
- `preserve l lf`, която за списък от цели числа `l` и списък от функции `lf` проверява дали всяка функция запазва всички елементи на списъка;
- `preservedby l lf`, която за списък от цели ч исла `l` и списък от функции `lf` връща списък с функциите, които запазват всички елементи на списъка;

Зад. 5 (15%, Дизайн на БД)

За система за продажда на автобусни билети се създава база данни.
В нея се съхранява информация за *автобусна гара* с уникален номер (число), име (низ до 100 символа, уникален), град, в който е гарата, и държава (един град може да има повече от 1 гари).
Пътуване се представя с уникален номер (число), начална и крайна гара, час на тръгване и час на пристигане, цена на билета (реално число с 2 знака след десетичната запетая).
Автобус се представя с уникален номер (число), регистрационен номер (до 8 символа, без интервали) и брой места (цяло число).
За всяко пътуване се задава поне 1 автобус. БД трябва да пази данни и за дата на пътуване на всеки автобус.
а) Да се направи E/R модел на базата данни.
б) Да се преобразува E/R моделът в релационен. Където е възможно, да се премахнат излишествата.
в) Да се напишат DDL команди, които реализират релационния модел и всички ограничения.

Зад. 6 (5-10%, БД - заявки)

Дадена е база данни за прегледи:



- Doctors съхранява информация за лекари:
  - Id е уникален номер, първичен ключ;
  - Name е фамилия на лекар;
  - Specialty е специалност на лекаря.
- Patients съхранява информация за пациенти:
  - Id е уникален номер, първичен ключ;
  - Name са имената на пациента;
  - Gender е пол на пациента ('f' - жена, 'm' - мъж);
  - DoctorId е външен ключ към личен лекар на пациента, може да е NULL.
- Records съхранява информация за извършени прегледи, първичният ключ е съставен от полетата DoctorId, PatientId, ExamDate:
  - DoctorId е уникален номер на лекара, извършил прегледа;
  - PatioentId е уникалният номер на прегледания пациент;
  - ExamDate е дата на извършване на прегледа;
  - Diagnose е диагнозата на пациента от прегледа.

Да се напишат следните SQL заявки:
а) извежда всички пациенти, които за прегледани от лекар с уникален номер 101, но за които не е личен лекар;
б) извежда справка във вида `Пациент | Лекар | Брой прегледи`, която дава информация за броя прегледи при различните лекари на пациент с номер 10101.

Зад. 7 (10%, АПИС: UC)

За описаната в зад. 5 система да се направи подробен потребителски случай за "закупуване на билет". Да включва основен сценарий, поне 2 алтернативни сценарии и нефункционални изисквания.

Зад. 8 (5-10%, АПИС: UML)

За описаната в зад. 5 система да се направи диаграма на състоянията на билет (резервиран, закупен и т.н.), като се обясни (обоснове).

Теория

I група: **7** и **23** (*Обектно ориентирано програмиране – Основни принципи. Класове и обекти. Оператори. Шаблони на функции и класове. Наследяване и полиморфизъм* и *Дискретни разпределения. Задачи, в които възникват. Моменти – математическо очакване и дисперсия.*)
II група: **5** и **11** (*Модели на разпределени ИТ архитектури. Среди и протоколи за разпределени приложения. Основни системни средства за планиране и управление на разпределената цифрова и информационна обработка.* и *Бази от данни. Релационен модел на данните.*)

Изтеглените теми показват, че **всяка може да се падне**, така че не разчитайте на информация кой е в комисията :)

Начало
Начална страница
Членове на сайта
Присъедини се
За автори
Препоръки за писане на статии
Формат на статиите
Последни промени
Панел за администратори
Често задавани въпроси
Лекции КН
Първи Курс
Втори курс
Трети курс
Четвърти курс
Лекции ПМ
Втори курс
Избираеми дисциплини
Математическа Логика
Дискретна Оптимизация
Теория на Множествата
Теория на Графите
Избрани Глави от Анализа
Държавен изпит
Информационни системи
Компютърни науки
Приложна математика

page revision: 6, last edited: 22 Jul 2013, 19:24 (2115 days ago)