Zadání a řešení testu z informatiky a zpráva

o výsledcích přijímacího řízení do magisterského navazujícího studia od podzimu 2015

Zpráva o výsledcích přijímacího řízení

do magisterského navazujícího studia od podzimu 2015

Počet podaných přihlášek	514
Počet přihlášených uchazečů	470
Počet uchazečů, kteří splnili podmínky přijetí	299
Počet uchazečů, kteří nesplnili podmínky přijetí	171
Počet uchazečů přijatých ke studiu, bez uvedení počtu uchazečů přijatých ke studiu až na základě výsledku přezkoumání původního rozhodnutí	299
Počet uchazečů přijatých celkem	299
Percentil pro přijetí	21,00

Základní statistické charakteristiky

		Informatika	Matematika	Celkem
Počet otázek		30	25	55
Počet uchazečů, kteří se zúčastnili přijímací zkoušky		239	237	239
Nejlepší možný výsledek		30.00	25.00	55.00
Nejlepší skutečně dosažený výsledek		26.75	25	48.5
Průměrný výsledek		15.55	11.58	27.04
Medián		16.75	11.25	27.75
Směrodatná odchylka		5.64	5.39	9.90
	Percentil			
Decilové hranice výsledku *	10	7.25	4.5	11.15
	20	10.75	7.25	19.5
	30	13.75	8.25	23.35
	40	15.25	9.75	25.75
	50	16.75	11.25	27.75
	60	17.5	13.25	30.45
	70	18.9	14.55	32.5
	80	20.35	16.5	35.25
	90	22	18.75	39.25

* Decilové hranice výsledku zkoušky vyjádřené d1, d2, d3, d4, d5, d6, d7, d8, d9 jsou hranice stanovené tak, že rozdělují uchazeče seřazené podle výsledku zkoušky do stejně velkých skupin, přičemž d5 je medián.

Přijímací zkouška - Informatika

Jméno a příjmení – pište do okénka	*Číslo přihlášky*	*Číslo zadání*
		15

Algoritmizace a datové struktury

---

1 Hašovací algoritmus používaný v databázích poskytuje přístup s časovou složitostí $O(1)$:

***A** k hodnotě záznamu na základě znalosti odpovídajícího klíče
B k hodnotě klíče na základě znalosti hodnoty záznamu
C k hodnotě všech obsažených záznamů bez znalosti klíče
D k hodnotě klíče X na základě znalosti klíče Y, pro který platí, že X = Y - 1
E nikoli, přístup ke klíči adresovanému pomocí hašovacího algoritmu má časovou složitost $\theta(log(n))$

---

2 Předpokládejme, že každé n-bitové kladné celé číslo X je uloženo jako oboustranně zřetězený seznam jednotlivých bitů čísla X. Prvním prvkem seznamu je nejméně významný bit čísla X. Např. číslo X = 12 v desítkové soustavě je uloženo jako seznam (0,0,1,1) délky 4. Jaká je časová složitost operace nahrazení libovolného čísla X uloženého v této datové struktuře za hodnotu X / 8, pokud použijeme celočíselné dělení?

***A** $\theta(1)$
B $\theta(log(n))$
C $\theta(n)$
D $\theta(n^2)$
E $\theta(n^3)$

---

3 Pro datovou strukturu AVL strom (samovyvažující se binární vyhledávací strom) platí:

A obsahuje kořen, uzly, listy a alespoň jeden cyklus
***B** jeho výška je vždy logaritmická vzhledem k počtu uzlů
C má-li $(n-1)$ uzlů, pak obsahuje právě n hran
D operace vyhledávání v AVL stromu s n uzly má časovou složitost $O(log(log(n)))$
E výška AVL stromu je vzhledem k počtu uzlů v nejhorším případě kvadratická

---

4 Pro datovou strukturu zásobník platí:

A struktura obsahuje vždy alespoň dva prvky (dno a vrchol zásobníku)
B naposledy vložený prvek bude odebrán jako poslední
***C** naposledy vložený prvek bude odebrán jako první
D prvek s nejmenší hodnotou je vždy udržován na dně zásobníku
E prvek s nejmenší hodnotou je vždy udržován na vrcholu zásobníku

---

5 Pro datovou strukturu kompletního binárního stromu platí, že každý uzel s výjimkou listů má právě dva potomky. Kolik interních uzlů (tj. uzlů mimo listy) je v kompletním binárním stromě obsahujícím 256 listů?

A 127
***B** 255
C 256
D 128
E 512

Počítačové systémy

---

6 Které číslo ve dvojkové soustavě je ekvivalentem čísla vyjádřeného v šestnáctkové (hexadecimální) soustavě jako A629?

***A** 1010 0110 0010 1001
B 1001 0010 0110 1010
C 1001 0100 0110 0101
D 0111 0110 0010 1001
E 42537

---

7 Příznak přenosu (carry flag) slouží v procesorech rodiny x86 k indikaci:

***A** přenosu z nejvýznamnějšího bitu
B liché parity výsledku operace
C nulového výsledku operace
D povolení obsluhy maskovatelného přerušení
E nutnosti obsloužit maskovatelné přerušení

---

8 Petersonův algoritmus lze typicky použít k:

***A** řešení problému vzájemného vyloučení
B plánování procesoru
C obsluze přerušení typu výpadek stránky
D hledání oběti pro uvolnění rámce při zaplnění fyzické paměti (při správě virtuální paměti)
E řešení problému typu výprask (thrashing)

---

9 Při potřebě alokace 100 B procesu P a minimální alokační jednotce operačního systému 4 kB dochází k:

***A** interní (vnitřní) fragmentaci paměti
B externí (vnější) fragmentaci paměti
C uváznutí (deadlock)
D vzájemnému vyloučení
E stárnutí

---

10 Nechť běh procesů na procesoru plánujeme algoritmem SJF (Shortest Job First) v preemptivní variantě (s předbíháním). Požadavek na proces P1 vzniká v čase 0 a proces potřebuje 11 jednotek procesoru, požadavek na proces P2 vzniká v čase 5 a proces potřebuje 2 jednotky procesoru, požadavek na proces P3 vzniká v čase 5 a proces potřebuje 3 jednotky procesoru, požadavek na proces P4 vzniká v čase 5 a proces potřebuje 4 jednotky procesoru. Plánování procesů začíná v čase 0. Jaký proces bude běžet na procesoru v čase 12?

A P1
B P2
C P3
***D** P4
E žádný z těchto procesů

Programování

---

11 Je dán programový zápis (úsek programu):

```
a = 0;
b = 5;
c = 0;
while (a < 5) {
   b = 5;
   while (b > a) {
      c = c + 1;
      b = b - 1;
   }
   a = a + 1;
}
```

Pro obsah proměnných a, b, c po konci uvedeného kódu bude platit:

A a = 0, b = 5, c = 0
***B** a = 5, b = 4, c = 15
C vykonání kódu neskončí, program cyklí
D a = 4, b = 5, c = 20
E a = 4, b = 4, c = 5

---

12 Je dán programový zápis (úsek programu):

```
index1 = 0;
index2 = 0;
while ((index1 < 10) && (index2 < 10)) {
   if (pole1[index1] < pole2[index2]) {
      PRINT(pole1[index1]);
      index1 = index1 + 1;
   }
   else {
      PRINT(pole2[index2]);
      index2 = index2 + 1;
   }
}
while (index1 < 10) {
    PRINT(pole1[index1]);
    index1 = index1 + 1;
}
while (index2 < 10) {
    PRINT(pole2[index2]);
    index2 = index2 + 1;
}
```

Předpokládejte, že `pole1` a `pole2` jsou souvislá pole o 10 prvcích seřazená vzestupně. Indexování polí je od 0, tj. první prvek je na pozici `pole1[0]`, druhý na `pole1[1]`, ... poslední desátý prvek na pozici `pole1[9]`. Funkce `PRINT` vypíše hodnotu poskytnutého prvku. Operátor `&&` značí logický součin (tj. splnění obou logických podmínek). Rozhodněte, která z uvedených možností je správná:

A Program vypíše pouze obsah `pole1`.
B Program vypíše pouze obsah `pole2`.
C Program vypíše obsah `pole1` a za ním obsah `pole2`.
***D** Program vypíše vzestupně seřazený spojený obsah polí `pole1` i `pole2`.
E Program vypíše sestupně seřazený spojený obsah polí `pole1` i `pole2`.

---

13 Rozhodněte, které z uvedených tvrzení je v běžných OOP jazycích (C++, Java, C#) platné:

A Blok, ve kterém se provádí kód, který může vyvolat výjimku, která je zachytávána, je obalen klauzulí catch {}.
B Pokud není výjimka obsloužena v aktuální funkci, tak se výjimka zahazuje a pokračuje se další instrukcí ve volající funkci.
C Po obsloužení výjimky se pokračuje v kódu na řádku následujícím po řádku, ve kterém došlo k vyvolání výjimky.
D Standardní knihovna neobsahuje žádné třídy pro výjimku – výjimky jsou vždy uživatelsky definované třídy.
***E** Pokud není výjimka obsloužena v aktuální funkci, tak se propaguje o úroveň výš do volající funkce.

---

14 Rozhodněte, které z uvedených tvrzení je v běžných OOP jazycích (C++, Java, C#) platné:

A Zapouzdření umožňuje skrýt vnitřní implementaci některých metod, neumožňuje ale skrýt atributy třídy.
B Prostřednictvím přístupnosti lze řídit přístup k atributům třídy, ale ne k jejím metodám.
C Zapouzdření pomáhá zjednodušit hlavičky metod skrytím všech jejich argumentů.
***D** Zapouzdření pomáhá zvyšovat robustnost implementace tím, že skrývá detaily vnitřního stavu objektu před okolím a omezuje možnost jeho změny.
E Zapouzdření je automaticky zajišťováno běhovým prostředím na úrovni ochrany paměti bez nutnosti specifikovat přístupnost.

---

15 Předpokládejte, že procesor provede volání funkce nebo metody a začne se vykonávat její tělo. Rozhodněte, která z uvedených možností (pro běžné jazyky typu C++, Java, C#) obsahuje právě všechna pravdivá tvrzení z možností I., II. a III.

I. Ukazatel na zásobník (stack pointer) je aktualizován.
II. Ukazatel na haldu (heap pointer) je aktualizován.
III. Čítač instrukcí (program counter) je aktualizován.

A I.
B II.
C I. a II.
***D** I. a III.
E II. a III.

Počítačové sítě

---

16 Při přenosu dat vznikají chyby. Samoopravné kódování dat přijímači umožní:

A všechny chyby opravit
***B** většinu chyb opravit, některé chyby pouze detekovat, výjimečně některé nezjistit
C většinu chyb detekovat
D většinu chyb opravit, ostatní chyby pouze detekovat
E většinu chyb opravit se znalostí tajného hesla, některé chyby nezjistit

---

17 Informace jsou digitální nebo analogové. Signály jsou digitální nebo analogové. Vyberte pravdivé tvrzení:

A Analogové signály mohou přenášet pouze analogové informace.
B Digitální signály vždy přenáší analogové informace bez zkreslení.
C Digitální i analogové signály mohou přenášet pouze digitální informace.
***D** Digitální i analogové signály mohou přenášet digitální i analogové informace.
E Digitální i analogové signály mohou přenášet pouze analogové informace.

---

18 Mezidoménové směrování s protokolem BGP (Border Gateway Protocol) charakterizujeme těmito vlastnostmi:

***A** Podporuje komplexní topologie, směrovače si vyměňují popis celých cest včetně skoků, používá CIDR pro agregaci cest, směruje podle politik.

B Podporuje komplexní topologie, směrovače si vyměňují informace o svých sousedech, pomalá konvergence způsobuje nekonzistenci ve směrovacích tabulkách.

C Směrovače si vyměňují kompletní kopie svých směrovacích tabulek, využívají metodu dělení horizontu.

D Směrovače využívají metriku tvořenou cenou cesty odvozenou od šířky pásma, provádějí load-balancing pro více cest se stejnou cenou.

E Podporuje komplexní topologie, směrovače si vyměňují popis celých cest včetně skoků, hledají optimalní řešení dle použité metriky.

---

19 Cílem adresace na transportní vrstvě síťového modelu OSI je

A propojit dva libovolné body globální sítě

B propojit dva libovolné body lokální sítě

***C** propojit odesílající a přijímající aplikaci

D odlišit MAC adresy všech zařízení v lokální síti

E propojit dva libovolné body senzorové sítě

---

20 Jmenná služba DNS zajišťuje:

A Překlad fyzických adres (MAC) na síťové a naopak. Pracuje na síťové vrstvě.

***B** Překlad doménových jmen na IP adresy a naopak. Je součástí aplikační vrstvy.

C Překlad doménových jmen na fyzické adresy a naopak. Je součástí vrstvy datového spoje.

D Překlad IPv4 adres na IPv6 adresy a naopak. Pracuje na síťové vrstvě.

E Překlad IPv4 adres a IPv6 adres na fyzické adresy.

Databázové systémy

---

21 Uvažujte následující relace z databáze banky: *zákazník(rč, jméno, adresa)*, *účet(č_účtu, rč, zůstatek)* a *úvěr(č_úvěru, rč, výše_úvěru, zaplaceno)*. Předpokládejte, že jeden zákazník může mít v bance více účtů i úvěrů, ale nemusí mít žádný. Z následujících SQL dotazů vyberte ten, který vrátí celkovou bilanci banky, tedy celkový objem peněz, které jsou na účtech zákazníků, ale které nejsou půjčeny někomu na úvěr:

***A** SELECT SUM(peníze) FROM (SELECT zůstatek AS peníze FROM účet UNION SELECT zaplaceno - výše_úvěru AS peníze FROM úvěr) celkem;

B SELECT SUM(peníze) FROM účet, úvěr WHERE účet.rč = úvěr.rč

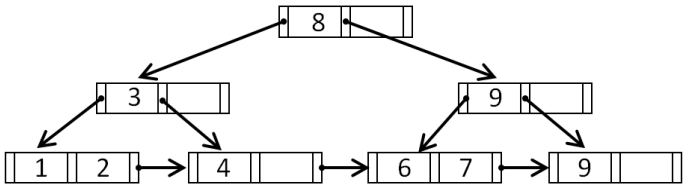
C SELECT SUM(zůstatek) + SUM(zaplaceno) - SUM(výše_úvěru) FROM účet, úvěr WHERE účet.rč = úvěr.rč

D SELECT SUM(zůstatek + zaplaceno - výše_úvěru) FROM účet NATURAL INNER JOIN úvěr

E (SELECT SUM(zůstatek) FROM účet) MINUS (SELECT SUM(zaplaceno - výše_úvěru) FROM úvěr)

---

22 Uvažujte následující B+strom, který tvoří index pro atribut id nějaké relace:



Jestliže databáze potřebuje zpřístupnit záznam s id = 3, pak:

***A** budou provedeny tři operace porovnání a záznam nebude nalezen
B budou provedeny dvě operace porovnání a záznam nebude nalezen
C bude provedena jedna operace porovnání a záznam nebude nalezen
D budou provedeny tři operace porovnání a záznam bude nalezen
E budou provedeny dvě operace porovnání a záznam bude nalezen

---

23 Která z následujících vlastností **není vyžadována** pro databázovou transakci?

A Atomicita - transakce se provede buď celá, nebo se neprovede vůbec.
B Konzistence - po provedení transakce zůstává databáze v konzistentním stavu.
C Izolovanost - operace prováděné uvnitř transakce jsou viditelné pouze pro tuto transakci.
D Trvalost - změny provedené dokončenou transakcí jsou uloženy v databázi a nemohou být ztraceny.
***E** Dokončitelnost - každá transakce musí vždy v konečném čase provést všechny požadované změny.

---

24 Vyberte pravdivé tvrzení o primárním klíči (PK):

A Pro jednu entitní množinu existuje několik PK.
B PK je maximální počet závislých atributů.
***C** PK je kandidátní klíč.
D PK má vždy minimálně dva atributy.
E PK je libovolná podmnožina atributů.

---

25 Uvažujme relace *zákazník(rč, jméno, adresa)* a *účet(č_účtu, rč, zůstatek)*. Vyberte, které z následujících tvrzení **není pravdivé**, pokud je atribut *účet.rč* nenulovým cizím klíčem (NOT NULL FOREIGN KEY):

A Relace zákazník může být prázdná.
B Relace účet může být prázdná.
***C** Relace účet obsahuje minimálně tolik záznamů, kolik je záznamů v relaci zákazník.
D Všechny hodnoty v atributu *účet.rč* existují v relaci zákazník.
E Některé hodnoty atributu *zákazník.rč* nemusí existovat v relaci účet.

Softwarové inženýrství

---

26 Jako softwarový inženýr hledáte programovací paradigma, které vám pomůže co nejlépe realizovat koncepty zapouzdření a polymorfismu. Které z následujících paradigmat má k těmto konceptům nejblíže (je jimi přímo charakterizováno)?

A Procedurální
***B** Objektově orientované
C Funkcionální
D Logické
E Datově orientované

---

27 Který z následujících popisů nejlépe charakterizuje techniku Test-Driven Development (TDD)?

A TDD je strategie testování softwaru, která vyzdvihuje význam manuálního testování před automatickým.

B TDD je strategie testování softwaru, která vyzdvihuje statické metody testování před dynamickými.

C TDD je přístup k vývoji softwaru, který doporučuje po implementaci každé části funkcionality napsat test a část otestovat.

***D** TDD je přístup k vývoji softwaru, který doporučuje v každém kroku implementace napsat test dříve než je daná část funkcionality implementována.

E TDD je přístup k vývoji softwaru, který doporučuje průběžné testování vyvíjeného softwaru vždy na konci každého měsíce.

---

28 Jako softwarový inženýr zvažujete, zda nově vznikající systém cílit k nasazení na jediný server nebo jej navrhnout jako distribuovaný (nasazený na více serverů). Uvažujme následující tři argumenty:

I. Pokud jsou výkonnostní charakteristiky řešení na jednom serveru vyhovující, dám přednost nasazení na jeden server, protože přechod do distribuovaného prostředí může přinést snížení spolehlivosti a bezpečnosti systému.

II. Pokud to je finančně únosné, dám vždy přednost distribuovanému řešení, protože je s ním spojena vyšší výkonnost, spolehlivost a bezpečnost.

III. Pokud to je finančně únosné, dám vždy přednost distribuovanému řešení, protože je s ním spojena vyšší výkonnost, bezpečnost a snažší udržovatelnost.

Rozhodněte, které z uvedených argumentů I., II. a III. jsou obecně validní (zvolte možnost obsahující všechny a právě všechny z nich):

A II.

B III.

***C** I.

D II. a III.

E I. a II.

---

29 Který z diagramů jazyka Unified Modelling Language (UML) zvolíte pro modelování procesů probíhajících v systému?

A Diagram datových toků (data flow diagram)

***B** Diagram aktivit (activity diagram)

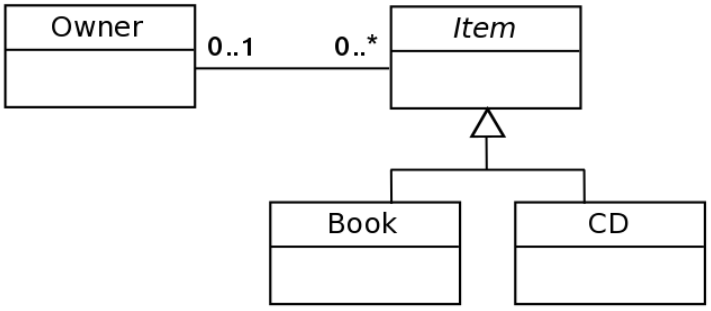
C Entitně relační diagram (entity-relationship diagram)

D Diagram následků (consequence diagram)

E Diagram tříd (class diagram)

---

30 Uvažujme návrh znázorněný následujícím diagramem tříd UML s vyznačením dědičnosti (vztah zakončený trojúhelníkem) a abstraktní třídou Item:



Na základě tohoto diagramu platí:

A Objekt typu Owner může mít libovolné množství referencí na instance třídy Item, ale nemůže mít referenci na instanci třídy CD nebo Book.
***B** Objekt typu Owner může mít libovolné množství referencí na instance třídy Book nebo CD, ale nemůže mít referenci na instanci třídy Item.
C Objekt typu Owner může mít libovolné množství referencí na instance třídy Book nebo Item.
D Objekt typu Owner může mít nejvýše jednu referenci na instanci třídy Item.
E Objekt typu Owner může mít nejvýše jednu referenci na instanci třídy Book, CD nebo Item.