Subiectele pentru proba practică din cadrul examenului de Paradigme de Programare Sesiunea iunie 2014

Considerații generale cu privire la proba practică:

La examen studentul va avea acces la cursurile de Paradigme de Programare și la documentația Java SE 6 care poate fi descărcată de la adresa: http://www.oracle.com/technetwork/java/javase/downloads/jdk-6u25-doc-download-355137.html.

Biletul va fi compus din două subiecte din lista de mai jos. Combinația de două subiecte va fi generată în mod aleatoriu. Timpul de rezolvare pentru ambele subiecte va fi de 2 ore.

Pentru a promova proba de laborator trebuie îndeplinite următoarele:

- Măcar una din problemele de pe bilet trebuie rezolvată complet în decursul celor două ore.
- Studentul trebuie să fie capabil să răspundă la întrebări cu privire la rezolvarea problemei.

În cazul în care studentul primește o notă mai mică ca 5 la proba de laborator nu va mai susține proba teoretică, examenul considerându-se picat.

Pentru studenții care au rezolvat o problemă complet se mai poate extinde cu maxim jumătate de ora examinarea practică în cazul în care mai au nevoie de timp ca să termine ce-a de-a doua problemă de pe bilet.

Lista de subiecte

1. Creați o ierarhie de clase în C++ formată din clasa abstractă Mamifer cu atributele tip (evil,divine) și data nașterii, și metodele: mănâncă, merge la baie, hrănește animalele. Să se deriveze o clasă Bunica și clasele PisicaDeCartier, PisicaSiameza și PisicaEgipteana. Acestea din urma mai au metoda miauna unde se aplica polimorfismul și fiecare miaună diferit. Acțiunile în acest caz sunt implementate prin mesaje afișate la consolă.

2. Să se scrie un program ASM care calculează factorialul și suma $\sum_{i=0}^{n} i$. Procesorul țintă va fi x286. Variabilele vor fi definite numai în memorie cu vizualizarea rezultatului în emulator.

3. Să se scrie un program ASM care calculează suma elementelor unui vector de întregi. Variabilele vor fi definite numai în memorie cu vizualizarea rezultatului în emulator.

4. Să se implementeze în ASM ordonarea unui vector folosind metoda swap din curs. Variabilele vor fi definite numai în memorie cu vizualizarea rezultatului în emulator. Se vor prezenta și diagramele de clase de bază.

5. Să se implementeze în ASM ordonarea unui vector folosind metoda insertion sort. Variabilele vor fi definite numai în memorie cu vizualizarea rezultatului în emulator. Se vor prezenta și diagramele de clase de bază.

6. Să se implementeze în C++ folosind template o stiva cu rezervarea dinamică a memoriei. Se vor prezenta și diagramele de clase de bază.

7. Să se implementeze în C++ folosind template o lista circulară cu rezervarea dinamică a memoriei. Se vor prezenta și diagramele de clase de bază.

8. Să se implementeze în C++ folosind template o lista de numere complexe precum și operatori supraîncărcați ca în curs, cu rezervarea dinamică a memoriei. Se vor prezenta și diagramele de clase de bază.

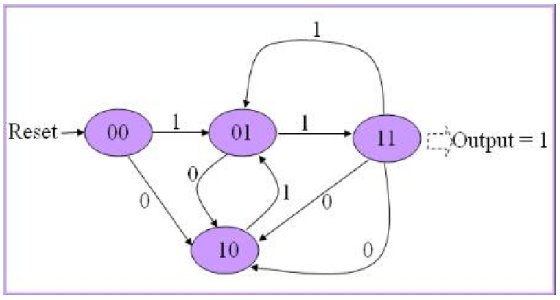
9. Să se implementeze în C++ problema hambarului din curs la care se adaugă câteva reguli de interacțiune între elementele din hambar: instanțieri aleatoare ale unui număr și tip de animale se executa/interacționează în clasa Curte pe baza regulilor.

10. Să se implementeze în Java folosind interfețe o stiva cu rezervarea dinamică a memoriei. Se vor prezenta și diagramele de clase de bază.

11. Să se implementeze în Java folosind interfețe o lista circulară cu rezervarea dinamică a memoriei. Se vor prezenta și diagramele de clase de bază.

12. Să se implementeze în Java aplicația ceasului digital după modelul din curs.

13. Să se implementeze în Java, folosind oricare din cele două metode prezentate la curs, un program care să implementeze funcționarea automatului prezentat mai jos prin graful său de fluența. Se vor prezenta și diagramele de clase de bază.



14. Să se scrie o aplicație în Java care definește o clasă Numărător care, la schimbarea valorii, generează un eveniment CounterEvent. Acest eveniment va fi receptat de o clasă Receptor care, la apariția evenimentelor, afișează valoarea numărătorului. Se vor prezenta și diagramele de clase de bază.

15. Să se scrie o aplicație Java folosind AWT care afișează o fereastră având o caseta text și trei butoane având etichetele 1,2,3. La apăsarea butoanelor în caseta text să apară eticheta butonului apăsat. Se vor prezenta și diagramele de clase de bază.

16. Să se scrie o aplicație Java folosind Swing care afișează o fereastră având o caseta text și trei butoane având etichetele 1,2,3. La apăsarea butoanelor în caseta text să apară eticheta butonului apăsat. Se vor prezenta și diagramele de clase de bază.

17. Să se scrie o aplicație Java folosind Swing care se rulează într-o fereastră și la apăsarea mouse-ului în zona ferestrei să apară o eticheta care să conțină coordonatele apăsării mouse-ului. Se vor prezenta și diagramele de clase de bază.

18. Să se scrie o aplicație Java folosind AWT care se rulează într-o fereastră și la apăsarea mouse-ului în zona ferestrei să apară o eticheta care să conțină coordonatele apăsării mouse-ului. Se vor prezenta și diagramele de clase de bază.

19. Să se scrie o aplicație Java care implementează în mod grafic folosind un Canvas jocul spânzurătoarea. Se va folosi un dicționar cu zece cuvinte definit intern sau preluat dintr-un fișier. Se vor prezenta și diagramele de clase de bază.

20. Să se scrie o aplicație Java care implementează în mod grafic folosind un JPanel jocul spânzurătoarea. Se va folosi un dicționar cu zece cuvinte definit intern sau preluat dintr-un fișier. Se vor prezenta și diagramele de clase de bază.

21. Să se scrie o aplicație Java care implementează în AWT un editor de tip Notepad (avem unde edita, bare de scroll și butoane control). Se vor prezenta și diagramele de clase de bază.

22. Să se scrie o aplicație Java care implementează în Swing un editor de tip Notepad (avem unde edita, bare de scroll și butoane control). Se vor prezenta și diagramele de clase de bază.

23. Pornind de la fabrica de forme prezentată în curs, să se construiască în Java o fabrică de nave spațiale: una pătrat, una triunghi și una cerc care să fie desenate pe Canvas. Se vor prezenta și diagramele de clase de bază.

24. Pornind de la fabrica de forme prezentata în curs, să se construiască în Java o fabrica de nave spațiale una pătrat, una triunghi și una cerc care să fie desenate pe JPanel. Se vor prezenta și diagramele de clase de bază.

25. Pornind de la fabrica de forme prezentata în curs, să se implementeze în Java pentru o navă pătrat desenată în Canvas, o fabrica de ”foc” în care Să se obțină trei tipuri de foc: linear (mitraliera), blast (o minge de foc) și rachete (mai multe linii). Se vor prezenta și diagramele de clase de bază.

26. Pornind de la exemplul din curs, să se creeze și să se testeze în Java (cu acțiuni modelate prin mesaje consolă) un Adapter între clasa Bunica și oricare din clasele Pisica, Caine și Purcel. Se vor prezenta și diagramele de clase de bază.

27. Să se creeze în Java o aplicație pentru rezervarea unei camera de hotel. Camerele (cu atributele și operatorii lor) sunt obiecte și sunt reținute într-un vector. Cu AspectJ se va implementa un flux suplimentar (după modelul din curs) de tratare a listei de așteptare în caz de cerere de rezervare care nu poate fi satisfăcută imediat. Se vor prezenta și diagramele de bază.

28. Să se creeze în Java o aplicație pentru rezervarea unei camera de hotel. Camerele (cu atributele și operatorii lor) sunt obiecte și sunt reținute într-un vector. Cu AspectJ se va implementa un flux suplimentar (după modelul din curs) de tratare plății următorului serviciu suplimentar: consum din barul camerei când se generează nota de plată. Se vor prezenta și diagramele de bază.

29. Să se creeze în Java o aplicație pentru rezervarea unei camera de hotel. Camerele (cu atributele și operatorii lor) sunt obiecte și sunt reținute într-un vector. Cu AspectJ se va implementa un flux suplimentar (după modelul din curs) de tratare plății următorului serviciu suplimentar: plata servicii SPA când se generează nota de plată. Se vor prezenta și diagramele de bază.

30. Să se implementeze un program Java care va implementa un scheduler pentru thread pool (ferma de fire de execuție) în conformitate cu algoritmul FCFS prezentat la curs.

31. Pornind de la exemplul de buffer implementat cu producător-consumator din curs (cu eliminarea blocajului) Să se implementeze în Java o coadă folosită în transmiterea și recepția de șiruri de caractere între mai multe thread-uri (un fel de chat unde fiecare client este pe câte un thread).

32. Să se construiască în Java un pool de thread-uri care să facă niște calcule simple (gen sume, inducție. etc) să pună în evidență hazardul de curse (vezi în curs).

33. Să se implementeze în Java un calcul $\sum_0^n i$ cu 4 thread-uri simultane care fiecare calculează pe un subinterval folosind modelul master/slave.

34. Să se proceseze în Java calculul $\sum_0^n i$ simultan pentru 4 valori diferite ale lui n luate dintr-o coadă de către 4 task-uri diferite (model peer).

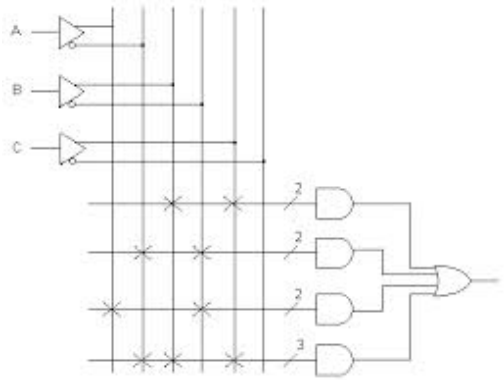
35. Să se realizeze în Java o procesare după model pipeline a unui tablou de întregi. Primul thread din pipe înmulțește toate elementele vectorului V cu o constantă alpha, următorul thread din pipe va ordona vectorul, iar final ultimul thread îl va afișa în coordonate x și y.

36. Să se implementeze în Java mai multe cozi de thread-uri (care execută în bucla calcule matematice complicate) cu priorități diferite și să se implementeze mecanismul de prevenire a înfometării (cel de îmbătrânire).

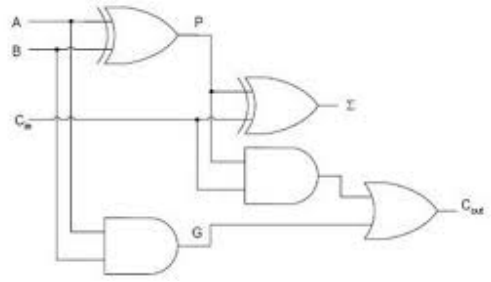
37. Pentru sincronizarea unor thread-uri, să se implementeze în Java protocolul cu simularea limitării priorității (highest locker).

38. Să se scrie în Java un program simplu cu thread-uri (pornind de la exemplul din curs) care să folosească atât lock-ul pe instanță cât și cel static.

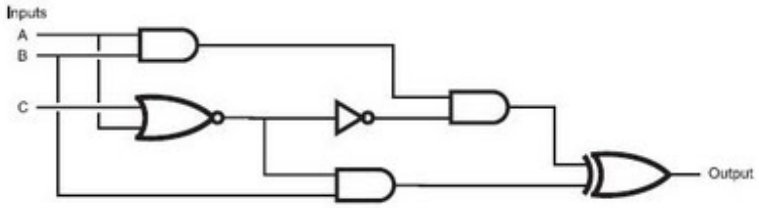
39. Pe baza exemplului de la curs (sursa modulului de simulator clc: openbookproject.net/py4fun/logic/logic.py), să se implementeze în Python calculul funcției din figura:



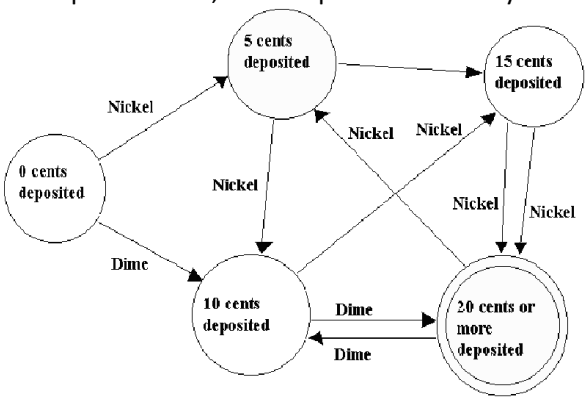
40. Pe baza exemplului de la curs (sursa modulului de simulator clc: openbookproject.net/py4fun/logic/logic.py), să se implementeze în Python calculul funcției din figura:



41. Pe baza exemplului de la curs (sursa modulului de simulator clc: openbookproject.net/py4fun/logic/logic.py), să se implementeze în Python calculul funcției din figura:



42. Folosind exemplul din curs, să se implementeze în Python următorul automat finit:



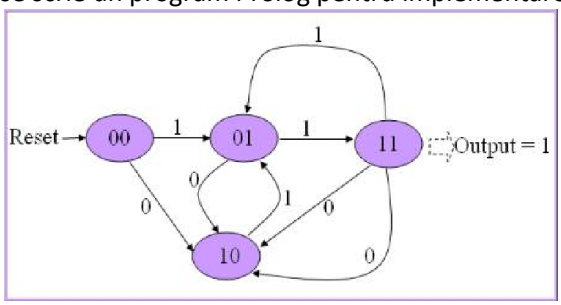
43. Să se creeze un program Python care primește un fișier oarecare (exe, dll, doc etc) la intrare și produce într-un fișier text cu același nume un hash cu md5 peste el.

44. Pe baza exemplului (folosind TCP) din curs. Să se realizeze în Python o aplicație de chat simplu între doi parteneri.

45. Propuneți și scrieți un program pentru o reprezentare pentru dreptunghi, pătrat și cerc ca o structura Prolog. (De exemplu un dreptunghi poate fi reprezentat de patru puncte)

46. Fie următoarele: big(bear). big(elephant). small(cat). brown(bear). black(cat). gray(elephant). dark(Z):- black(Z). dark(Z):- brown(Z). în care din cele doua cazuri Prolog trebuie să muncească mai mult înainte de a găsi răspunsul ? ?- big(X), dark(X). sau ?- dark(X), big(X).

47. Să se scrie un program Prolog pentru implementarea următorului automat:



48. Să se scrie un program Prolog care să poată găsi ultimul element dintr-o listă (?- my_last(X,[a,b,c,d]). → X = d).

49. Să se scrie un program Prolog care să poată elimina duplicatele dintr-o listă fără a schimba ordinea acestora (?- compress([a,a,a,a,b,c,c,a,a,d,e,e,e,e],X). → X = [a,b,c,a,d,e]).

50. Să se scrie un program Prolog care să poată elimina fiecare al n-lea element dintr-o listă (?- drop([a,b,c,d,e,f,g,h,i,k],3,X). → X = [a,b,d,e,g,h,k]).