FIŞA DISCIPLINEI

1. Date despre program

1.1 Instituţia de învăţământ superior	Universitatea Tehnică din Cluj-Napoca
1.2 Facultatea	Automatică şi Calculatoare
1.3 Departamentul	Calculatoare
1.4 Domeniul de studii	Calculatoare si Tehnologia Informatiei
1.5 Ciclul de studii	Licenţă
1.6 Programul de studii / Calificarea	Calculatoare si Tehnologia Informatiei / Inginer
1.7 Forma de învăţământ	IF – învăţământ cu frecvenţă
1.8 Codul disciplinei	32.

2. Date despre disciplină

2.1 Denumirea disciplinei	*Programare funcţională*				
2.2 Titularii de curs	Conf. dr.ing. Radu Razvan Slavescu Radu.Razvan.Slavescu@cs.utcluj.ro				
2.3 Titularul/Titularii activităţilor de seminar/laborator/proiect	Prof.dr.ing. Camelia Pintea Ing. Luminița Marghescu Ing. Vlad Petrean Ing. Florin Lele Ing. Irina Petrea				
2.4 Anul de studiu	3	2.5 Semestrul	5	2.6 Tipul de evaluare *(E – examen, C – colocviu, V – verificare)*	E
2.7 Regimul disciplinei	*DF – fundamentală, DD – în domeniu, DS – de specialitate, DC – complementară*				DD
	DI – Impusă, DOp – opțională, DFac – facultativă				DI

3. Timpul total estimat

3.1 Număr de ore pe săptămână	4	din care:	Curs	2	Seminar		Laborator	2	Proiect	
3.2 Număr de ore pe semestru	56	din care:	Curs	28	Seminar		Laborator	28	Proiect	
3.3 Distribuţia fondului de timp (ore pe semestru) pentru:										
(a) Studiul după manual, suport de curs, bibliografie şi notiţe										18
(b) Documentare suplimentară în bibliotecă, pe platforme electronice de specialitate şi pe teren										10
(c) Pregătire seminarii / laboratoare, teme, referate, portofolii şi eseuri										10
(d) Tutoriat										4
(e) Examinări										2
(f) Alte activităţi:										0
3.4 Total ore studiu individual (suma (3.3(a)…3.3(f)))						44				
3.5 Total ore pe semestru (3.2+3.4)						100				
3.6 Numărul de credite						5				

4. Precondiţii (acolo unde este cazul)

4.1 de curriculum	Cursul de Structuri de Date și Algoritmi
4.2 de competenţe	Nu sunt necesare cunostinte anterioare de Programare Functionala, dar este recomandat sa existe o experienta de minimun un an de programare intr-un limbaj cum ar fi Java, C sau C++

5. Condiţii (acolo unde este cazul)

5.1. de desfăşurare a cursului	Tabla, proiector, calculator
5.2. de desfăşurare a laboratorului	Calculatoare, interpretoare/compilatoare pentru limbajele studiate Prezenţă obligatorie 100% pentru admiterea la examenul final

6. Competenţele specifice acumulate

6.1 Competenţe profesionale	**C2 -** Proiectarea unui sistem software în maniera functionala **C2.1 – I**dentificarea și descrierea componentelor software ale sistemului **C2.2 -** Explicarea rolului fiecare componente și a interactiunii acestora **C2.3 -** Construirea unor componente software folosind metode de proiectare,

	limbaje, tehnologii și instrumente specifice Programării Functionale **C2.4 -** Implementarea componentelor software pe baza de cod în stil functional, idiomatic și eficient **C2.5 -** Evaluarea caracteristicilor funcţionale şi nefuncţionale ale sistemului software pe baza unor metrici de performanța și demonstrarea corectitudinii sale
6.2 Competenţe transversale	N/A

7. Obiectivele disciplinei

7.1 Obiectivul general al disciplinei	Dezvoltarea abilitatii de scrie cod corect, concis, pe baza elementelor paradigmei functionale (imutabilitatea variabilelor, nivel ridicat de abstractizare, posibilitatea demostrarii formale a corectitudiniii unui program, posibilitatea paralelizarii facile a codului), precum și a fundamentelor sale teoretice (calcul lambda)
7.2 Obiectivele specifice	Pentru atingerea acestor obiective generale, studenţii vor invata: - sa scrie cod în maniera functionala, cu eliminarea variabilelor de stare - sa identifice avantajele si dezavantajele diferitelor stiluri de programare - sa utilizeze recursivitatea și sa structureze codul pentru optimizarea sa - sa folosească functiile de ordin superior - sa exploateze mecanismele evaluarii lenese și structurile infinite - sa demonstreze formal corectitudinea unui program - sa manipuleze expresii lambda de baza

8. Conţinuturi

8.1 Curs	Nr.ore	Metode de predare	Observaţii
Introducere. Paradigme de programare. Noţiuni fundamentale de programare în Haskell, Elm: funcţii, identificatori, recursivitate	2		
Noţiuni fundamentale: recursivitate, constante, tipuri de date primitive, tuple, operatori infix, evaluare .	2	(Fizic) Slideuri, Demonstratii si reprezentare modele pe tabla, exercitii rapide pentru cresterea interactiunii	
Noţiuni fundamentale: declaratii locale, tipuri de date, polimorfism.	2		
Liste: construire listă, operaţii fundamentale pe liste.	2		
Liste: operatori de liste (generatori, garzi, extensiune listă).	2		
Arbori: date alternative, potrivire forme, excepţii, arbori binari, conversie arbori-liste.	2		
Arbori: arbori binari de căutare, verificare proprietate arbori echilibraţi AVL, printare.	2		
Implementare operaţii pe mulţimi. Rationator propozitional.	2		
Funcţii de ordin superior: funcţii anonime, aplicare parţială, relația dintre funcţii si date, combinatori	2		
Funcţii de ordin superior pentru liste (map, filter, fold)	2		
Date infinite: evaluare leneşă, obiecte infinite, structuri circulare.	2		
Calcul Lambda: notaţie lambda, conversii, combinatori.	2		
Raţionare asupra corectitudinii programelor: inducţie structurală, echivalenţa funcţiilor, inducţie pe numărul de noduri.	2		
Monade. Exemple de cazuri de utilizare	2		

Bibliografie *(bibliografia minimală a disciplinei conținând cel puțin o lucrare bibliografică de referință a disciplinei, care există la dispoziţia studenţilor într-un număr de exemplare corespunzător)*

1. Haskell - A Purely Functional Language, www.haskell.org
2. Elm – A Delightful language for reliable web applications, elm-lang.org
3. G. Hutton. **Programming in Haskell, 2nd edition** Cambridge University Press, 2016
4. M. Lipovaca. **Learn You a Haskell for Great Good.** No Starch Press, 2011.
5. I.A. Letia, L.A. Negrescu, L. Negrescu. **Programare functionala, vol. I.** Ed. Albastra, Cluj-Napoca, 2006

8.2 Aplicații (seminar/laborator/proiect)*	Nr.ore	Metode de predare	Observaţii
Introducere în Programarea Functionala folosind Elm	2		
Tipurin în Elm	2		
Liste și recursivitate	2	(Fizic)	

Funcţii de ordin superior în Elm	2	Rezolvare de exerciții și probleme, implementare de funcții pe calculator, trasare de algoritmi	
Evaluare Elm	2		
Miniaplicatie Elm	2		
Introducere în Haskell. Liste, recursivitate	2		
Verificare de tipuri în Haskell	2		
Arbori în Haskell	2		
Funcţii de ordin superior Haskell	2		
Evaluare leneşă și liste infinite	2		
Miniaplicatie în Haskell	2		
Calcul lambda	2		
Evaluare Haskell	2		

Bibliografie *(bibliografia minimală pentru aplicații conținând cel puțin o lucrare bibliografică de referință a disciplinei care există la dispoziţia studenţilor într-un număr de exemplare corespunzător)*

1. www.haskell.org
2. elm-lang.org
3. M. Lipovaca. **Learn You a Haskell for Great Good.** No Starch Press, 2011.

* *Se vor preciza, după caz: tematica seminariilor, lucrările de laborator, tematica și etapele proiectului.*

9. Coroborarea conţinuturilor disciplinei cu aşteptările reprezentanţilor comunităţii epistemice, asociaţiilor profesionale şi angajatorilor reprezentativi din domeniul aferent programului

Continutul disciplinei este in concordanta cu continutul unor cursuri similare ale altor universitati (Paradigme de Programare, Haskell paralel si concurent). Cursul evidentiaza tehnicile specifice programarii functionale care au patruns si patrund in limbajele (multiparadigma) moderne, precum si posibilitarea de a demonstra formal corectitudinea programelor. Studentii sunt incurajati sa identifice idei specifice Programarii Functionale prezente in practica curenta a companiilor locale IT.

10. Evaluare

Tip activitate	Criterii de evaluare	Metode de evaluare	Pondere din nota finală
Curs	Intelegerea conceptelor specifice paradigmei de programare functionala și bazei sale teoretice. Participare activa la curs, teme.	(Fizic) Examen final scris / Test Moodle	50%
Seminar			
Laborator	Cantitatea și calitatea codului Haskell și Elm dezvoltat Abilititatea de a identifica si corecta erori de programare.	(Fizic) Teste de laborator și mini-aplicatii	50%
Proiect			

Standard minim de performanţă: Înțelegerea și abilitatea de a dezvolta cod in limbaje functionale, folosid recursivitatea, functiile de ordin superior, potrivirea sabloanelor
Nota finala: 50% laborator + 50% examen scris
Conditii pentru participare la examenul final: Medie Laborator ≥ 5
Conditii pentru promovare: Medie Examen ≥ 5

Data completării:	Titulari	Titlu Prenume NUME	Semnătura
20.09.2022	Curs	Conf. dr.ing. Radu Razvan Slavescu	
	Aplicații	Prof.dr.ing. Camelia Pintea	
		Ing. Luminița Marghescu	
		Ing. Vlad Petrean	
		Ing. Florin Lele	
		Ing. Irina Petrea	

Data avizării în Consiliul Departamentului Calculatoare 19.09.2017	Director Departament Prof.dr.ing. Rodica Potolea
Data aprobării în Consiliul Facultății de Automatică și Calculatoare 20.09.2017	Decan Prof.dr.ing. Liviu Miclea